Journal of Artificial Intelligence Research 59 (2017) 815 – 848 Submitted 03/17; published 08/17

Complexity of n-Queens Completion

Ian P. Gent IAN.GENT@ST-ANDREWS.AC.UK
Christopher Jefferson CAJ21@ST-ANDREWS.AC.UK
Peter Nightingale PWN1@ST-ANDREWS.AC.UK
School of Computer Science, University of St Andrews,
St Andrews, Fife KY16 9SX, UK

Abstract

The n-Queens problem is to place n chess queens on an n by n chessboard so that no two queens are on the same row, column or diagonal. The n-Queens Completion problem is a variant, dating to 1850, in which some queens are already placed and the solver is asked to place the rest, if possible. We show that n-Queens Completion is both NP-Complete and #P-Complete. A corollary is that any non-attacking arrangement of queens can be included as a part of a solution to a larger n-Queens problem. We introduce generators of random instances for n-Queens Completion and the closely related Blocked n-Queens and Excluded Diagonals Problem. We describe three solvers for these problems, and empirically analyse the hardness of randomly generated instances. For Blocked n-Queens and the Excluded Diagonals Problem, we show the existence of a phase transition associated with hard instances as has been seen in other NP-Complete problems, but a natural generator for n-Queens Completion did not generate consistently hard instances. The significance of this work is that the n-Queens problem has been very widely used as a benchmark in Artificial Intelligence, but conclusions on it are often disputable because of the simple complexity of the decision problem. Our results give alternative benchmarks which are hard theoretically and empirically, but for which solving techniques designed for n-Queens need minimal or no change.

1. Introduction

The n-Queens problem is to place n chess queens on an n by n chessboard so that no two queens are on the same row, column or diagonal. This puzzle dates to 1848, and only two years later a variant was introduced by Nauck (1850) in which some number of queens are pre-placed and the solver is asked to place the rest, if possible. This is the n-Queens Completion problem and Figure 1 shows the first known instance studied. We will show that the n-Queens Completion problem is NP-Complete and #P-Complete, discuss solvers for the problem, and empirically analyse randomly generated instances. The n-Queens Completion problem may be one of the simplest NP-Complete problems to explain to people who understand the rules of chess. The problem is “Given an $n \times n$ chessboard on which some queens are already placed, can you place a queen in every remaining row so that no two queens attack each other?”

The n-Queens problem has an extraordinary history for such an apparently unassuming problem, both generally and inside Artificial Intelligence. Formerly, and incorrectly, attributed to Gauss, the problem's history was clarified by Campbell (1977). The 8-Queens problem was introduced by Bezzel (1848) and by Nauck (1850) (possibly independently). The latter publication attracted the interest of Gauss, who even made a small mistake in

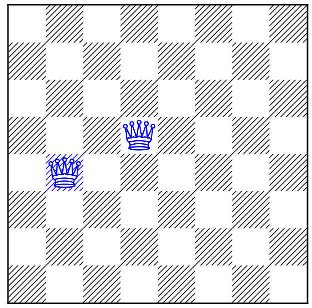


Figure 1: This is the first published instance of the n-Queens Completion problem, by Nauck (1850). The reader may enjoy attempting to place 6 more queens on the chessboard so that no two queens attack each other. Is it possible? If so, how many different ways are there to do it? The answers to these questions are given below in Figure 12.

studying the problem.[1] The generalisation to n-Queens has attracted the interest of many other mathematicians, whose results are surveyed by Bell and Stevens (2009). The first paper to describe backtracking search on computer, presented in 1958, does so for the n-Queens problem (Walker, 1960). Since then it has been used as an example and/or benchmark problem in many classic AI papers, for example at least six with more than 400 citations on Google Scholar at time of writing (Golomb & Baumert, 1965; Bitner & Reingold, 1975; Mackworth & Freuder, 1985; Minton, Johnston, Philips, & Laird, 1992; Selman, Levesque, & Mitchell, 1992; Crawford, Ginsberg, Luks, & Roy, 1996).

The complexity of the n-Queens problem is often misunderstood. The decision problem is solvable in constant time since there is a solution for all $n > 3$ so is only NP-hard if P=NP. A witnessing solution can be constructed easily (Bell & Stevens, 2009) but note that the witness (a set of n queens) requires $n \log n$ bits to specify but this is not polynomial in the size of the input, which is only $\log n$ bits. The n-Queens problem has often been incorrectly called NP-hard, even in well-cited papers (Mandziuk, 1995; Martinjak & Golub, 2007; Shah-Hosseini, 2009; Nakaguchi, Kenya, & Tanaka, 1999, each with at least 29 citations). The counting version of the problem, i.e. to determine how many solutions to n-Queens there are, is sequence A000170 of the Online Encyclopedia of Integer Sequences (Sloane, 2016). The sequence is currently known only to $n = 27$, for which the number of solutions is more than 2.34×10^{17}. No approach better than optimised exhaustive search seems to be known: e.g. the $n = 27$ total was counted using a massively parallel search using FPGAs (Preußer, 2016). Hsiang, Hsu, and Shieh (2004) show that solving the n-Queens counting problem is "beyond the #P-class". Bell and Stevens (2009) states that this means that there is no closed form expression in n for the number of solutions, but Chaiken, Hanusa, and Zaslavsky (2015)

1. He reported finding 76 solutions but later realised that four of those were erroneous so had only 72.

claim to give one.[2] Cadoli and Schaerf (2006) studied the n-Queens Completion problem without studying its decision or counting complexity. The most closely related work to ours is by Martin (2007), who proves a rather different generalisation of n-Queens to be NP-complete, the key difference being that some squares can be marked as stopping attacks. In particular, this means that solutions to Martin's problem need not be solutions to the n-Queens problem, unlike n-Queens Completion.

We are contributing to a rich literature on the complexity of puzzles and games. One of the earliest results in the area is that generalised chess is EXPTIME-complete (Fraenkel & Lichtenstein, 1981). Amongst other games to have been proved NP-complete or harder are the card solitaire games Klondike (Longpré & McKenzie, 2009), Freecell (Helmert, 2003) and Black Hole (Gent, Jefferson, Kelsey, Lynce, Miguel, Nightingale, Smith, & Tarim, 2007), the Sudoku puzzle (Takayuki & Takahiro, 2003), video games like Pac-Man (Viglietta, 2014), and casual games such as Minesweeper (Kaye, 2000), Candy Crush Saga and Bejeweled (Walsh, 2014; Guala, Leucci, & Natale, 2014). Many other games are surveyed by Kendall, Parkes, and Spoerer (2008) and by Demaine and Hearn (2009). In each case, the complexity-theorist must define a generalised class of instances. Unlike some of the examples above, this step is completely natural for n-Queens Completion.

Because of the ease of finding a solution, the n-Queens problem has been subject to repeated controversy in AI over whether it should be used as a benchmark at all. For example a sequence of papers argued the point in the pages of SIGART Bulletin in the early 1990s (Sosic & Gu, 1990; Johnson, 1991; Bernhardsson, 1991; Gu, 1991; Valtorta, 1991), and then in 2014 the issue was raised again in a blog post by Smet (2014). We resolve this issue in the sense that, as an NP- and #P-Complete problem, n-Queens Completion does provide a valid benchmark problem. Similarly, the Quasigroup Completion Problem (which is to complete a partially filled latin square) is NP-Complete (Colbourn, 1984) and is a challenging and popular benchmark (Gomes & Selman, 1997), whereas constructing a latin square from scratch is trivial. A very closely related problem, "Blocked n-Queens", has previously been used for benchmarking without complexity guarantees (Namasivayam & Truszczynski, 2009). Our results show as a corollary that Blocked n-Queens is NP-Complete and #P-Complete. We explore the practical difficulty of these problems, and a new variant, the Excluded Diagonals Problem. For Blocked n-Queens and the Excluded Diagonals Problem, we show the existence of a phase transition associated with hard instances, but we were not able to generate consistently hard instances for the n-Queens Completion problem.

The remainder of this paper is structured as follows. We introduce a number of basic terms and concepts in Section 2. We give definitions of a sequence of problems in Section 3, where the first problem is n-Queens Completion, and the last is a variant of Boolean Satisfiability (SAT) that is NP- and #P-Complete, with the Excluded Diagonals Problem at an intermediate stage. The proof proceeds by a sequence of polynomial reductions in Section 4, starting with the last problem and ending with n-Queens Completion. Finally, we present an empirical study of random n-Queens Completion, Blocked n-Queens, and Excluded Diagonals Problems in Section 5.

2. It is unclear to us if this is a mathematical dispute or simply a dispute on what it means to be a closed form expression, but in any case Chaiken et al.'s formula has not been used to extend knowledge of the number of solutions beyond $n = 27$.

2. Background and Definitions

We will represent a queen as an ordered pair of integers. Since these represent coordinate positions in a grid, we will use vector notation where appropriate, although we will not use any significant vector algebra. Given a queen represented by an ordered pair (α, β), the value α represents the queen's column, and β its row on the chessboard. The values $\alpha + \beta$ and $\alpha - \beta$ represent the two diagonals the queen is on. It may not be intuitively obvious that chessboard diagonals correspond to sums and differences, but consider moving one square along the two orthogonal diagonals: in one direction the sum of the coordinates does not change, while in the other direction the difference does not change. To avoid possible confusion about which direction is which, we follow Bell and Stevens (2009) in simply calling $\alpha + \beta$ the 'sum-diagonal' and $\alpha - \beta$ the 'difference-diagonal'.

Definition 1. *A* queen $\mathbf{q}$ *is a pair of integers* (α, β) *with* $0 \leq \alpha$, $0 \leq \beta$. *For the rest of these definitions we assume* $\mathbf{q} = (\alpha, \beta)$ *and where appropriate a second queen* $\mathbf{q}_1 = (\alpha_1, \beta_1)$.

- *The* "column of $\mathbf{q}$", *written* $\text{col}(\mathbf{q})$, *is* α. *The* "row of $\mathbf{q}$", *written* $\text{row}(\mathbf{q})$, *is* β. *The* "sum-diagonal of $\mathbf{q}$" *is* $\alpha + \beta$ *or equivalently* $\text{col}(\mathbf{q}) + \text{row}(\mathbf{q})$. *The* "difference-diagonal of $\mathbf{q}$" *is* $\alpha - \beta$ *or equivalently* $\text{col}(\mathbf{q}) - \text{row}(\mathbf{q})$.

- *The* "size of $\mathbf{q}$", *written* $\|\mathbf{q}\|$ *is defined by* $\|\mathbf{q}\| \stackrel{def}{=} \max(\alpha, \beta) = \max(\text{col}(\mathbf{q}), \text{row}(\mathbf{q}))$. *We say that* $\mathbf{q}$ "fits on board size n" *iff* $\|\mathbf{q}\| < n$. *We extend this to sets of queens, so that* $\|Q\| \stackrel{def}{=} \max_{\mathbf{q} \in Q}(\|\mathbf{q}\|)$. *As normal, we write* $|Q|$ *for the number of elements in* Q.

- *We say that* "$\mathbf{q}$ attacks $\mathbf{q}_1$" *if they are not equal and any of the row/column/sum-diagonal/difference-diagonal is the same in* $\mathbf{q}$ *and* $\mathbf{q}_1$; *i.e.* $\mathbf{q}$ *attacks* $\mathbf{q}_1$ *iff* $\mathbf{q} \neq \mathbf{q}_1$ *and any of* $\alpha = \alpha_1$, $\beta = \beta_1$, $\alpha + \beta = \alpha_1 + \beta_1$, *or* $\alpha - \beta = \alpha_1 - \beta_1$.

- *Given an integer* i, *we write* $i\mathbf{q} \stackrel{def}{=} (i\alpha, i\beta)$ *for the product of* i *and* $\mathbf{q}$. *We write* $\mathbf{q} + \mathbf{q}_1 \stackrel{def}{=} (\alpha + \alpha_1, \beta + \beta_1)$. *We extend these notations to sets of queens* Q *and* Q_1 *in the natural ways:* $iQ \stackrel{def}{=} \{i\mathbf{q} \mid \mathbf{q} \in Q\}$, $Q + \mathbf{q}_1 \stackrel{def}{=} \{\mathbf{q} + \mathbf{q}_1 \mid \mathbf{q} \in Q\}$, $Q + Q_1 \stackrel{def}{=} \{\mathbf{q} + \mathbf{q}_1 \mid \mathbf{q} \in Q, \mathbf{q}_1 \in Q_1\}$.

Problem 1 (n-Queens)**.** Problem*:* $\langle n \rangle$ *where* n *is an integer* ≥ 1. Solution*: A set* Q *of queens which all fit on board size* n *such that:* $|Q| = n$; *and for any two distinct queens* $\mathbf{q}_1, \mathbf{q}_2 \in Q$, $\mathbf{q}_1$ *does not attack* $\mathbf{q}_2$. *If, additionally, for any two distinct queens* $(\alpha_1, \beta_1), (\alpha_2, \beta_2) \in Q$, *both* $\alpha_1 + \beta_1 \neq \alpha_2 + \beta_2 \bmod n$ *and* $\alpha_1 - \beta_1 \neq \alpha_2 - \beta_2 \bmod n$, *then* Q *is said to be a* "modular" *solution to the* n*-Queens problem.*

The modular n-Queens problem can be thought of as the n-Queens with the top and bottom edges of the board identified, and similarly the left and right edges identified, so the board forms a torus. Bell and Stevens (2009, Thm. 6) give the following result of Pólya. In terms of chessboards it says that we can obtain a solution to the $n \times m$-Queens by taking a standard m-Queens solution and replacing each empty square with an $n \times n$ empty board, and replacing each queen with a copy of a solution to the modular n-Queens problem. It can be expressed in our terms as follows.

Theorem 2 (Pólya)**.** *If we have a modular solution Q_1 to the n-Queens problem, and any solution Q_2 to the m-Queens problem, then $Q_1 + nQ_2$ is a solution to the $n \times m$-Queens problem.*

In Theorem 2 the same modular solution is n-embedded at each one of m positions. We will need to embed distinct sets of queens to create one board so we define *embedding* at a single position.

Definition 3. *Given any queen $\mathbf{q}_1$, a set of queens Q_2, and a natural number n (where $\|Q_2\| < n$), the n-embedding of Q_2 at position q_1 is defined as $n\mathbf{q}_1 + Q_2$.*

3. A Sequence of Problems

In order to prove that n-Queens Completion is both NP-Complete and #P-Complete, we will use a sequence of *parsimonious reductions* as described by Papadimitriou (1994, Chapter 18). A parsimonious reduction maps instances of one problem to another such that the number of solutions is exactly preserved and the reduction can be computed in polynomial time.

In this section we define n-Queens Completion and three increasingly general problems that generalise n-Queens Completion. We also define a restricted form of 1-in-3-SAT that is NP-Complete and #P-Complete. The proof (in the following section) is a sequence of four parsimonious reductions, starting with restricted 1-in-3-SAT and ending with n-Queens Completion.

For each problem we define the parameters and the space of solutions. There is deliberately no distinction made between the decision form of the problem (does a solution exist?) and the counting form (how many solutions are there?). When it is important, the distinction will usually be clear from context (for example, whether we are discussing NP-Completeness or #P-Completeness).

Problem 2 (**n-Queens Completion**)**.** PROBLEM*: $M_2 = \langle n, P\rangle$ where n is an integer and P is a set of queens that all fit on board size n and no two queens in P have the same column or row as each other.* SOLUTION*: A set S_2 of queens which is a solution to the n-Queens Problem for n and such that $P \subseteq S_2$.*

Note that this definition of n-Queens Completion does *not* require the input set P of queens to be mutually non-attacking, but does insist that they are all on separate rows and columns. This choice makes some cases in our later proofs easier, but makes no significant difference to solving complexity: if two queens in P attack each other on a diagonal, the answer 'no' can be given trivially.

We next generalise n-Queens Completion by allowing restrictions that queens may only appear in a subset of the rows and columns (and must appear in each specified row and column).

Problem 3. PROBLEM*: $M_3 = \langle n, P, C, R\rangle$ where $\langle n, P\rangle$ is an instance of Problem 2, and C, R are sets of integers such that $|R| = |C|$ and for any queen $(\alpha, \beta) \in P$, $\alpha \notin C$ and $\beta \notin R$.* SOLUTION*: A set S_3 of queens such that: $|S_3| = |C| + |P|$; $P \subseteq S_3$; no two queens in S_3 attack each other; and for any pair $(\alpha, \beta) \in S_3 \setminus P$ we have $\alpha \in C$, $\beta \in R$.*

The reason that we exclude elements of P from C and R is to simplify later proofs. We next generalise Problem 3 by giving sets of sum-and difference-diagonals that must not appear. Unlike the row and column restrictions of Problem 3, there may be diagonals which are permitted but contain no queen in a given solution.

Problem 4 (**Excluded Diagonals**). Problem*:* $M_4 = \langle n, C, R, D^-, D^+\rangle$*, where* $M_3 = \langle n, \{\}, C, R\rangle$ *is an instance of Problem 3 and* D^-, D^+ *are sets of integers with* $D^- \subseteq \{-(n-1), \dots, n-1\}$*,* $D^+ \subseteq \{0, \dots, 2n-2\}$*.* Solution*: A set of queens* S_4 *which is a solution to* M_3 *and additionally: for any queen* $(\alpha, \beta) \in S_4$ *we have* $\alpha - \beta \notin D^-$*,* $\alpha + \beta \notin D^+$*.*[3]

Each of the problems above has a board size parameter n. We assume that the size of the encoding of a problem instance to a Turing machine input has size bounded below by n and polynomial in n.

Our next problem class contains a set of instances of Problem 4, and ignores constraints between problems except for the sum-diagonals. Two queens placed within different sub-problems in the set are allowed to have the same row, column, or difference-diagonal, but *not* the same sum-diagonal. Thus we can compose instances of Problem 4 with control over their interactions.

Problem 5. Problem*: A set* $M_5 = \{M_{4,a} \mid 0 \le a < |M_5|\}$*, where each* $M_{4,a}$ *is an instance of Problem 4 with* $D^+ = \{\}$*.* Solution*: A set* $S_5 = \{S_{4,a} \mid 0 \le a < |M_5|\}$*, where each* $S_{4,a}$ *is a solution to* $M_{4,a}$*, and additionally: for any* $\{S_{4,a}, S_{4,b}\} \subseteq S_5$*, and any* $(\alpha_a, \beta_a) \in S_{4,a}$*,* $(\alpha_b, \beta_b) \in S_{4,b}$*, we have* $\alpha_a + \beta_a \neq \alpha_b + \beta_b$*.*

We will use a restriction of the 1-in-3-SAT problem, in which all literals are positive, no variable occurs in the same clause twice, and no variable occurs in more than three clauses.

Problem 6 (Restricted 1-in-3-SAT)**.** Problem*: A pair* $M_6 = \langle V, C\rangle$ *where* C *is a set (of clauses) such that each* $c \in C$ *is a set of size 3 of variables,* $c = \{v_i, v_j, v_k\}$*; and where* $V = \{v \mid \exists c \in C \,.\, v \in c\}$ *is the set of all variables that are contained in any clause. Each variable* $v \in V$ *occurs in at most three clauses in* C*.* Solution*: A truth assignment* $S_6 : V \to \{true, false\}$ *such that for all* $c = \{v_i, v_j, v_k\} \in C$*,* S_6 *maps exactly one of* v_i, v_j, v_k *to true and the other two to false.*

Theorem 4. *Restricted 1-in-3-SAT is NP-Complete and #P-Complete.*

Proof. Porschen, Schmidt, Speckenmeyer, and Wotzlaw (2014, Lemma 4) prove that this problem is NP-Complete. The #P-Completeness of 1-in-3-SAT limited to positive literals (Monotone 1-in-3-SAT) is a corollary of Creignou and Hermann's (1996) general results. This does not apply to the restriction on variable occurrences, so we must prove that Restricted 1-in-3-SAT is #P-Complete.

We will give a parsimonious reduction of Monotone 1-in-3-SAT to Restricted 1-in-3-SAT. Let n_x be the number of occurrences of variable x. We will create new variables that we

3. Note that the sets D^-, D^+ are diagonals that may not be used, while the sets R and C are rows and columns that must be used. This disparity is because we find ourselves specifying which columns are allowed (and implicitly the rest are disallowed), while usually specifying which diagonals are disallowed (and implicitly the rest are allowed).

force to be true and false, specifically $\{t^x_i, f^x_{3i-2}, f^x_{3i-1}, f^x_{3i} \mid i = 1 \ldots \lceil 2n_x/3 \rceil\}$. For each i we add the three clauses

$$\{t^x_i, f^x_{3i-2}, f^x_{3i-1}\}, \{t^x_i, f^x_{3i-1}, f^x_{3i}\}, \{t^x_i, f^x_{3i}, f^x_{3i-2}\}$$

If some t^x_i were false, then exactly three variable occurrences in the set of pairs $\{f^x_{3i-2}, f^x_{3i-1}\}$, $\{f^x_{3i-1}, f^x_{3i}\}$ and $\{f^x_{3i}, f^x_{3i-2}\}$ would have to be true, which is impossible since each literal occurs twice. So these clauses force each t^x to be true and each f^x to be false, using only three occurrences of each t variable and two of each f. For each occurrence of variable x in the original clause set we introduce two new variables x_i and $\bar{x}_i$. We create the clauses:

$$\{x_1, \bar{x}_1, f^x_1\}, \{x_2, \bar{x}_2, f^x_2\}, \ldots, \{x_{n_x}, \bar{x}_{n_x}, f^x_{n_x}\},$$

$$\{\bar{x}_1, x_2, f^x_{n_x+1}\}, \{\bar{x}_2, x_3, f^x_{n_x+2}\}, \ldots, \{\bar{x}_{n_x-1}, x_{n_x}, f^x_{2n_x-1}\}$$

Since each f is false, the first line of clauses ensures that each x_i and $\bar{x}_i$ take opposite values, while the second line of clauses ensures that all variables x_i take the same value. Note that the clauses use at most two occurrences of each x_i and $\bar{x}_i$, leaving one x_i to be used for a positive occurrence of x in the original clause set. We construct a clause set by replacing the i^{th} (necessarily positive) occurrence of variable x in the original clause set with the variable x_i, and then adding the clauses introduced above. A solution in which all x_i are true corresponds with x being true in the original, while a solution in which all x_i are false corresponds with x being false. This reduction is parsimonious because all values of x_i and $\bar{x}_i$ in the new problem are uniquely determined by the value of x in the original while all t variables are true and all f variables false. This uses at most three occurrences of each new variable and all are positive. Finally note that the translated problem has no variable repeated in a single clause: this is by inspection in all the new clauses, and for the translated versions of original clauses, note that every literal in every clause is replaced by a different variable. □

4. A Sequence of Reductions

We will give a series of reductions from Problem 6 to Problem 2. Each reduction will be polynomial. Each will also be parsimonious, i.e. the number of solutions to the reduced problem will be the same as the number of solutions to the original, thus simultaneously proving Problem 2 to be NP-complete and #P-complete. If we were interested in NP-completeness alone, we could use a simpler non-parsimonious reduction from 3-SAT to Problem 5 (as described in Section 4.4), but we are not aware of a way to simplify the sequence of reductions from Problem 5 to Problem 2. Therefore the proof of NP-completeness alone would be almost as complex as the proof of both properties together.

4.1 Filling in the Gaps: Reduction from Problem 3 to n-Queens Completion

First we define sets of queens Q_{19}, A_n, B_n for use later in the reduction.

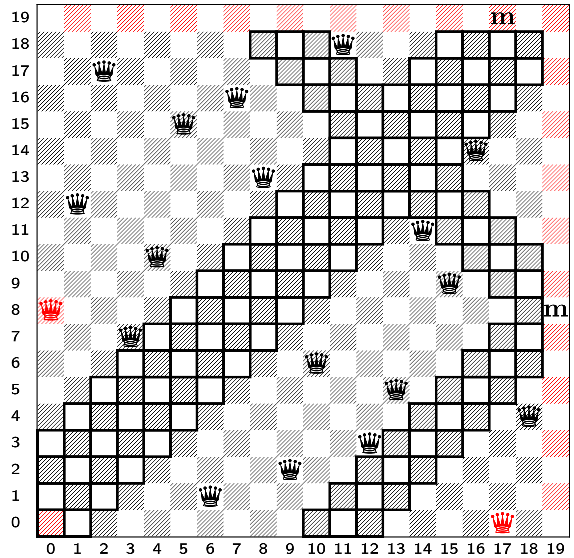


Figure 2: The solution Q_{19} to the 19-Queens problem with an additional column and row marked in red. Two squares (17,19) and (19,8) on the additional column and row are marked with ‘m’, and are not on the same or adjacent diagonals to each other. Also emphasised are squares (0,0), (0,8) and (17,0). A number of squares are outlined to show that no queen is placed on the diagonal (0,0) to (18,18) or either adjacent diagonal and that no queen is on the diagonals or adjacent diagonals of squares (17,19) and (19,8).

Definition 5. *We define the following sets of queens, illustrated in Figures 2 and 3.*

$$Q_{19} \stackrel{def}{=} \{(0,8),(1,12),(2,17),(3,7),(4,10),(5,15),(6,1),(7,16),(8,13),(9,2),\\ (10,6),(11,18),(12,3),(13,5),(14,11),(15,9),(16,14),(17,0),(18,4)\}$$

$$A_n \stackrel{def}{=} \{(i, 2i \bmod n) \mid 0 \leq i < n\}$$

$$B_n \stackrel{def}{=} \{(2i \bmod n, i) \mid 0 \leq i < n\}$$

Next we observe some properties of Q_{19} that will be used in later proofs.

Observation 6. *Q_{19} is a solution to the 19-Queens problem. It has no queen on difference-diagonals 1, 0 or -1. For the queens $(0, y), (x, 0)$, consider the pairs $(19, y)$ and $(x, 19)$: that is, in Q_{19} consider $(19, 8)$ and $(17, 19)$ (marked with ‘m’ in Figure 2). Their sum-/difference-diagonals are not within 1 of each other, $(0, 0)$ or any queen in Q_{19}.*

All these properties can be checked by inspection. They can also be visually seen in Figure 2. Q_{19} was found by a search using Savile Row (Nightingale, Akgün, Gent, Jefferson,

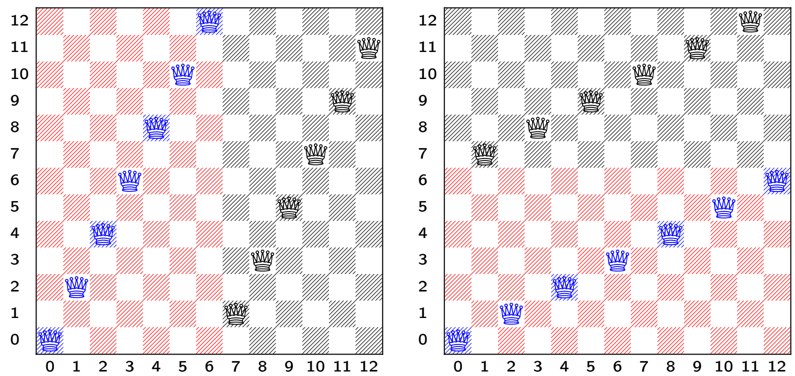


Figure 3: Two modular solutions A_{13} (left) and B_{13} (right) to the 13-Queens problem as defined in Definition 5 with each queen in A_{13} at $(i, 2i \mod 13)$ and in B_{13} at $(2i \mod 13, i)$ (right.) The first seven columns on the left have no two queens in adjacent rows, while the first seven rows on the right have no two queens in adjacent columns. The two solutions are mirror images along the diagonal (0,0)-(12,12), and also use exactly the same sets of both sum- and difference-diagonals.

Miguel, & Spracklen, 2017) and Minion (Gent, Jefferson, & Miguel, 2006) from a model available online.[4] Further Savile Row/Minion searches show that there is no $n < 19$ for which a solution to n-Queens can have all the properties of Observation 6.

Lemma 7. *If $n = 1 \mod 6$ or $n = 5 \mod 6$ then A_n and B_n are modular solutions to the n-Queens problem, and use the same sets of sum-/difference-diagonals as each other.*

Proof. Bell and Stevens (2009, Thm. 9) prove that A_n is a modular solution to the n-queens problem for $n = 1 \mod 6$ or $n = 5 \mod 6$. As B_n is symmetrically equivalent to A_n (reflected about the longest difference-diagonal), this implies B_n is modular as well.

Because $(x, y) \in A_n$ iff $(y, x) \in B_n$, the sets of sum-diagonals in A_n and B_n are the same. As n is odd, the difference-diagonals of A_n are $\{i - 2i \mid i \in \{1..\frac{n-1}{2}\}\} \cup \{i - (2i - n) \mid i \in \{\frac{n+1}{2}..n\}\}$, which simplifies to $\{-i \mid i \in \{1..\frac{n-1}{2}\}\} \cup \{n - i \mid i \in \{\frac{n+1}{2}..n\}\} = \{-\frac{n-1}{2}..\frac{n-1}{2}\}$. Because $(x, y) \in A_n$ iff $(y, x) \in B_n$ the difference-diagonals of A_n are the negation of the difference-diagonals of B_n, therefore the difference-diagonals of A_n and B_n are the same. □

Lemma 8 (Rivin and Zabih)**.** *Suppose we have a solution Q to the n-Queens problem, a subset of queens $P \subseteq Q$, and a set of queens P_1 which uses the same set of rows, columns, sum-diagonals and difference-diagonals as P. Then $(Q \setminus P) \cup P_1$ is also a solution to the n-Queens problem.*

Proof. This result is due to Rivin and Zabih (1992, Thm. 2.4). Since P_1 uses the same rows, columns and diagonals as P, no queen in $Q \setminus P$ can attack any queen in P_1. □

4. `https://ipg.host.cs.st-andrews.ac.uk/n-queens-completion-experiments-code.tgz`

The following lemma proves that if two queens are not in the same row, not in the same column, not on the same diagonal, and not on two adjacent diagonals in the same direction, then: if we multiply these two queens by n and add to each one any queen that fits on board size n then the resulting two queens will not attack each other.

Lemma 9. *Consider any two queens $\mathbf{q}_1, \mathbf{q}_2$ which both fit on board size n, and two queens $\mathbf{q}_3, \mathbf{q}_4$ of any size. Then: if the columns (respectively rows) of $\mathbf{q}_3, \mathbf{q}_4$ are different, then the columns (rows) of $\mathbf{q}_1 + n\mathbf{q}_3$ and $\mathbf{q}_2 + n\mathbf{q}_4$ are different. Also: if the sum-diagonals (respectively difference-diagonals) of $\mathbf{q}_3, \mathbf{q}_4$ differ by more than one, then the sum-diagonals (difference-diagonals) of $\mathbf{q}_1 + n\mathbf{q}_3$ and $\mathbf{q}_2 + n\mathbf{q}_4$ are different. If all the just-mentioned properties hold for $\mathbf{q}_3, \mathbf{q}_4$ then $\mathbf{q}_1 + n\mathbf{q}_3$ does not attack $\mathbf{q}_2 + n\mathbf{q}_4$.*

Proof. Let each $\mathbf{q}_i = (\alpha_i, \beta_i)$. For columns, suppose $\alpha_3 \neq \alpha_4$. Since $0 \leq \alpha_1, \alpha_2 < n$, $\alpha_1 + n\alpha_3 \neq \alpha_2 + n\alpha_4$, so the columns of $\mathbf{q}_1 + n\mathbf{q}_3$ and $q_2 + n\mathbf{q}_4$ are different. The case of rows is similar. For sum-diagonals, suppose that $|(\alpha_3 + \beta_3) - (\alpha_4 + \beta_4)| \geq 2$. The range of sum-diagonals on a board of size n is from 0 to $2n - 2$, so $|(\alpha_1 + \beta_1) - (\alpha_2 + \beta_2)| < 2n$. But $|n(\alpha_3 + \beta_3) - n(\alpha_4 + \beta_4)| \geq 2n$. Hence the sum-diagonals of $\mathbf{q}_1 + n\mathbf{q}_3$ and $q_2 + n\mathbf{q}4$ are different. The case of difference-diagonals is similar, since their range is $-(n - 1)$ to $n - 1$, so $|(\alpha_1 - \beta_2) - (\alpha_1 - \beta_2)| < 2n$. If all these cases apply then $\mathbf{q}_1 + n\mathbf{q}_3$ does not attack $\mathbf{q}_2 + n\mathbf{q}_4$, by definition of not attacking. □

We can now provide a reduction from Problem 3 to Problem 2. This reduction takes an instance of Problem 3 and n'-embeds it at location $(0, 0)$ of a large board based on Figure 2 (where n' is defined as part of the construction). The other queens in Figure 2 are also replaced with chessboards of queens, which ensure the problem is a valid instance of Problem 2 and block the rows and columns required of the copy of Problem 3 in the bottom-left corner. To solve the constructed instance of Problem 2, queens must be placed within the embedded instance of Problem 3 and nowhere else, hence a solution of Problem 2 maps back to Problem 3 very easily.

Definition 10 (Reduction from Problem 3 to Problem 2)**.** *Given an instance of Problem 3, $M_3 = \langle n_3, P_3, C, R\rangle$, we construct an instance of Problem 2, $M_2 = \langle n_2, P_2\rangle$ as follows:*

- *If a pair of queens $\{\mathbf{q}_1, \mathbf{q}_2\} \subseteq P_3$ attack each other, then we simply drop R and C and set $n_2 = n_3$, $P_2 = P_3$. Otherwise:*

1. *If $n_3 \bmod 3 = 2$ then set $n' = 2n_3 + 1$, otherwise set $n' = 2n_3 - 1$, so Lemma 7 may be applied.*
2. *Form the set of queens $Q_1 = A_{n'} + n'(Q_{19} \setminus \{(17, 0)\})$. Set $Q_2 = Q_1 \cup (B_{n'} + n'(17, 0))$.* That is, n'-embed a copy of $A_{n'}$ at position $\mathbf{q}$ for every $\mathbf{q} \in Q_{19}$, except for queen $(17, 0)$, for which we n'-embed a copy of $B_{n'}$.
3. *Set $C' = C \cup \{\mathrm{col}(\mathbf{q}) \mid \mathbf{q} \in P_3\}$, $R' = R \cup \{\mathrm{row}(\mathbf{q}) \mid \mathbf{q} \in P_3\}$. Set $Q_3 = \{\mathbf{q} \mid \mathbf{q} \in Q_2, \mathrm{col}(\mathbf{q}) \notin C', \mathrm{row}(\mathbf{q}) \notin R'\}$.* That is, we remove queens from Q_2 if they are in one of the columns in C, rows in R, or one of the same rows or columns as a queen in P_3. This can only remove queens from the copy of $A_{n'}$ that is n'-embedded at $(0, 8)$ or the copy of $B_{n'}$ that is n'-embedded at $(17, 0)$ in Q_2.

4. *Put the columns in C' in ascending order $\alpha_0 < \alpha_1 < \ldots < \alpha_{|C'|-1}$. Form the set of queens $Q_C = \{(i, 2\alpha_i) \mid 0 \leq i < |C'|\}$. Similarly order the rows $\beta_0 < \beta_1 < \ldots < \beta_{|R'|-1}$ in R' and set $Q_R = \{(2\beta_i, i) \mid 0 \leq i < |R'|\}$.*

5. *Set $n_2 = 19n' + |C'|, P_2 = Q_3 \cup (Q_C + n'(19, 8)) \cup (Q_R + n'(17, 19)) \cup P_3$.*

We will now prove that the construction in Definition 10 is correct, and explicitly show how to map from solutions of Problem 2 to Problem 3 and vice-versa.

Lemma 11. *If S_2 is a solution to M_2, then $S_3 = \{\mathbf{q} \mid \mathbf{q} \in S_2 \;\&\; \|\mathbf{q}\| < n_3\}$ is a solution to M_3.*

Proof. No queens in S_3 attack each other since S_2 is a solution. Certainly $P_3 \subseteq P_2$ by construction and $P_2 \subseteq S_2$ since S_2 is a solution. Since $\|P_3\| < n_3$ this gives us $P_3 \subseteq S_3$. It remains to show that $|S_3| = |C| + |P_3|$, and for every $\mathbf{q} \in S_3 \setminus P_3$, both $\mathrm{col}(\mathbf{q}) \in C$ and $\mathrm{row}(\mathbf{q}) \in R$. For a given column $\alpha \in C$, we have $\alpha < n_3$, and there must be $\mathbf{q}_\alpha \in S_2$ with $\mathrm{col}(\mathbf{q}_\alpha) = \alpha$, and this has $\mathrm{row}(\mathbf{q}_\alpha) = \beta + jn'$ for some $0 \leq \beta < n'$ and $0 \leq j \leq 19$. We will show that $j = 0$ and $\beta \in R$. The only possible nonzero value of j is 8, since all other rows are excluded by copies of $A_{n'}$: the possible values of β for $j = 8$ are the rows occupied by queens deleted at step 3, i.e. the rows $\{2\alpha \mid \alpha \in C'\}$ by construction of $A_{n'}$. But each of these is excluded by one of the queens in $Q_C + n'(19, 8)$ added at step 5, specifically by $(i, 2\alpha_i) + n'(19, 8)$ if α_i is the i^{th} element in C' in ascending order. So we have $j = 0$. To check $\beta \in R$, first note that all rows outside R' are excluded by queens from $B_{n'} + n'(17, 0)$, and rows from queens in P_3 are excluded by $P_3 \subseteq P_2$. So $\beta \in R' \setminus \{\mathrm{row}(\mathbf{q}) \mid \mathbf{q} \in P_3\} = R$. That is, we have shown that for all columns $\alpha \in C$ there is a queen $(\alpha, \beta) \in S_3$ with $\beta \in R$. This gives $|S_3| \geq |C| + |P_3|$. But $n' - (|C| + |P_3|)$ possible columns are ruled out by queens remaining from $A_{n'} + n'(0, 8)$, so $|S_3| \leq n' - (n' - (|C| + |P_3|)) = |C| + |P_3|$. So $|S_3| = |C| + |P_3|$ as required, and no queens in $S_3 \setminus P_3$ can use a column not in C or row not in R. $\square$

Lemma 12. *If S_3 is a solution to M_3, then $S_2 = S_3 \cup P_2$ is a solution to M_2.*

Proof. We note that $S_2 = S_3 \cup (Q_C + n'(19, 8)) \cup (Q_R + n'(17, 19)) \cup Q_3$. We will prove that no two queens within one of these four subsets attack each other, and that no queen in one subset attacks any queen in the other subsets.

First we consider attacks within one of the subsets. S_3 is a solution to M_3, so no two queens in it attack each other. By Lemma 7, $A_{n'}$ is a modular solution to n'-Queens, so by Theorem 2, Q_1 is a subset of a solution to $19n'$-Queens ($Q_1 \cup A_{n'} + n'(17, 0)$ is a solution to $19n'$-Queens). Lemma 7 applies since n' is constructed to be $= 1$ or $= 5$ mod 6, and with Lemma 8 shows that Q_2 is a solution to $19n'$-Queens, and $Q_3 \subseteq Q_2$, so no queens in Q_3 can attack each other. Attacks within $Q_C + n'(19, 8)$ are equivalent to attacks within Q_C. The queens in Q_C are $(i, 2\alpha_i)$, with $i = 0, 1, 2, \ldots$ and α_i strictly increasing with i. Queens within Q_C are trivially on different rows and columns; the sum-diagonal $i + 2\alpha_i$ is strictly increasing with i; and finally $2\alpha_{i+1} - 2\alpha_i \geq 2$ so $i + 1 - 2\alpha_{i+1} < i - 2\alpha_i$, so the difference-diagonals are strictly decreasing with i. So no queens within Q_C attack each other. The case of $Q_R + n'(17, 19)$ is similar, with the queens being $\{(2\beta_i, i) \mid 0 \leq i < |R'|\}$ and β_i strictly increasing.

Now we consider attacks between queens in different subsets. By the properties of Q_{19} in Observation 6, and from Lemma 9, no queen in $Q_C + n'(19, 8)$ attacks any queen in the rest of S_2 except possibly queens in $A_{n'} + n'(0, 8)$. But any queen in $A_{n'} + n'(0, 8)$ on the same row as a queen in $Q_C + n'(19, 8)$ is not in Q_3 by step 3. The case of $Q_R + n'(17, 19)$ is similar. The only remaining case is a queen in S_3 attacking a queen in Q_3. Again, we use Observation 6 and Lemma 9 to show that no queen in S_3 can be on the same diagonal as any queen in Q_3. Finally, all queens in S_3 must be in a column in C' and a row in R', but the queens in those columns and rows were removed at step 3 to form Q_3, so no attack along a row or column is possible. □

Theorem 13. *The reduction of Definition 10 is a parsimonious reduction from Problem 3 to Problem 2.*

Proof. From Lemma 11 and Lemma 12, and the fact that the maps defined in those lemmas are inverses of each other, since if $S_2 = S_3 \cup P_2$, we have $S_3 = \{\mathbf{q} \mid \mathbf{q} \in S_2 \; \& \; \|\mathbf{q}\| < n_3\}$. □

4.1.1 Including Arbitrary n-Queens Partial Solutions

Lemma 12 has simple corollaries that show *any* set of nonattacking queens can be found as a part of a solution to a larger n-Queens problem. While a brief detour from our complexity results, we find the result attractive and it may help illuminate the structure of the n-Queens problem. For example, it shows that *no* local arrangement of nonattacking queens can cause global unsolvability for all n. First we define inclusion of a set of queens P within a larger set Q. Inclusion is an embedding with the additional requirement that no queens that are *not* part of P occur in that subgrid of Q.

Definition 14. *If P is a set of queens with $\|P\| < n$, we say that P is n-*included *at location (x, y) in a set of queens Q if $P + (x, y) \subseteq Q$* ***and*** *there does not exist a queen $\mathbf{q}$ with $\|\mathbf{q}\| < n$ such that $\mathbf{q} \notin P$ and $\mathbf{q} + (x, y) \in Q$.*

Corollary 15. *For any set of queens P, where $n = \|P\|$, with no two queens in P attacking each other, P may be n-included at $(0, 0)$ in a solution of the n_2-Queens problem for some n_2.*

Proof. Set $M_3 = \langle n, P, \{\}, \{\}\rangle$, and construct an n-Queens Completion problem $M_2 = \langle n_2, P_2\rangle$ as in Definition 10. P is a solution of M_3, therefore (by Lemma 12) P_2 is a solution of M_2 that n-includes P at position $(0, 0)$. All solutions of $M_2 = \langle n_2, P_2\rangle$ are also solutions of the n-Queens problem $M_1 = \langle n_2\rangle$ so the corollary holds. □

Not only can we include P at $(0, 0)$, it can also be included at an arbitrary position on a larger grid simply by applying Corollary 15 to $P + (i, j)$ for any position (i, j). Furthermore, we can include as many copies of P as we want in a single solution to a larger n-Queens, and do so in exponentially many different ways. First, consider the four sets $P + (0, 0)$, $P + (0, 1)$, $P + (1, 0)$, $P + (1, 1)$. Assuming P is non-empty, these four sets are distinct and all include P. Second, consider the set of k queens $K = \{(3i, i) \mid 0 \le i < k\}$. The queens in K are all on different rows and columns and their sum-diagonals all differ by more than one from each other, as do their difference-diagonals. By Lemma 9 we can $(n+1)$-include any one of the four sets that include P in positions in $(n + 1)K$, giving a larger set of queens

P'. There are 4^k ways of constructing P'. By applying Corollary 15 to P' we have 4^k ways of including k copies of P in a solution to a larger n-Queens problem.

4.2 Ruling out Diagonals: Reduction from Problem 4 to Problem 3

Definition 16 (Reduction from Problem 4 to Problem 3)**.** *For the instance of Problem 4* $M_4 = \langle n, C, R, D^-, D^+\rangle$, *set* $Q_{D^-} = \{(6n-3-d, 3n-1-2d) \mid d \in D^-\}$, *set* $Q_{D^+} = \{(2n-2-d, n+2d) \mid d \in D^+\}$, *and set* $M_3 = \langle 7n-3, Q_{D^-} \cup Q_{D^+}, \{c+3n-2 \mid c \in C\}, R\rangle$.

The reduction is illustrated in Figure 4. The key point of this construction is that the difference-diagonal of $(6n-3-d, 3n-1-2d)$ is $3n-2+d$, and the sum-diagonal of $(2n-2-d, n+2d)$ is $3n-2+d$. The offset of $3n-2$ in both cases explains why we shift the set C by $3n-2$ in M_3.

First we confirm that M_3 is a well-defined instance of Problem 3, assuming M_4 is itself well-defined. The range of possible values of D^- is $-(n-1)\dots n-1$, and hence the extreme possible placements for queens in Q_{D^-} are $(7n-4, 5n-3)$ and $(5n-2, n+1)$; D^+ ranges from 0 to $2n-2$, so the extreme possible placements for queens in Q_{D^+} are $(0, 5n-4)$ and $(2n-2, n)$. These certainly have no overlap with the sets R and $\{c+3n-2 \mid c \in C\}$, and all fit on the board size $7n-3$ of M_3. No queen in $Q_{D^-} \cup Q_{D^+}$ is on the same row or column as any other: the only non-trivial observation necessary is that if n is odd (even), then queens in Q_{D^-} are all on even (odd) rows and Q_{D^+} on odd (even) rows. So M_3 is well-defined.

Lemma 17. *If* S_4 *is a solution to* M_4*, then* $S_3 = (S_4 + (3n-2, 0)) \cup Q_{D^-} \cup Q_{D^+}$ *is a solution to* M_3.

Proof. Within each subset of S_3, no pair of queens attack each other. In the case of $S_4 + (3n-2, 0)$ this is because S_4 is a solution, and in the other subsets by construction. We also noted above that queens in Q_{D^-} and Q_{D^+} are on different rows from each other, and their columns and diagonals are far apart. So we only need to consider potential attacks between $Q_{D^-} \cup Q_{D^+}$ and $S_4 + (3n-2, 0)$. But if $(\alpha, \beta) \in S_4$ then $\alpha - \beta \notin D^-$ and so $(6n-3-(\alpha-\beta), 3n-1-2(\alpha-\beta)) \notin Q_{D^-}$, therefore $(\alpha+3n-2, \beta)$ is not attacked by any queen in Q_{D^-}. Similar reasoning applies for Q_{D^+}. Finally we note that constraints on columns and rows in S_3 are obeyed because they are in M_4. □

Lemma 18. *If* S_3 *is a solution to* M_3*, then* $S_4 = \{(\alpha-3n+2, \beta) \mid (\alpha, \beta) \in S_3, \beta < n\}$ *is a solution to* M_4.

Proof. No two queens in S_4 attack each other since S_3 is a solution. They all come from the first n rows of S_3, so the range of $\alpha - 3n + 2$ is 0 to $n-1$, so $\|S_3\| < n$. The constraints on membership of C and R are satisfied since they are in M_3. For any $d \in D^-$, a queen in Q_{D^-} is on difference-diagonal $3n-2+d$, so no queen in S_3 is on that diagonal, hence no queen in S_4 is on difference-diagonal d. Similarly for diagonals in D^+, so the diagonal constraints in M_4 are obeyed. □

Theorem 19. *The reduction of Definition 16 is a parsimonious reduction from Problem 4 to Problem 3.*

Proof. From Lemma 17 and Lemma 18, and the fact that the maps defined in those lemmas are inverses of each other. □

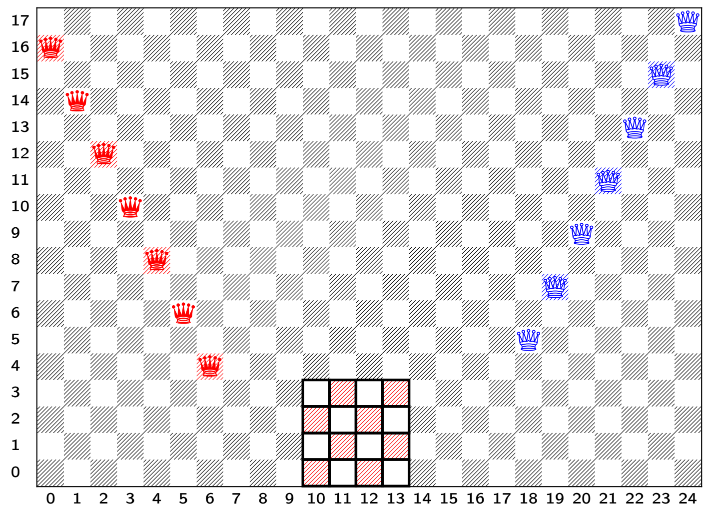


Figure 4: Ruling out the diagonals of the 4×4 square in bottom centre of the grid by Definition 16. A $n \times n$ square has $2n - 1$ diagonals in each direction. In this example we see 7 queens from (0,16) to (6,4) ruling out the sum-diagonals, and 7 queens from (18,5) to (24,17) ruling out the difference-diagonals. For example, the queen at (4,8) rules out the sum-diagonal 12, while the queen at (21,11) rules out the difference-diagonal 10. Note that none of these queens attack any squares in the 4×4 square except those in the intended diagonal, and also that no two of the 14 queens attack each other. Therefore we can use any subset of these 14 queens to rule out exactly and only a selected set of diagonals in the 4×4 square.

4.3 Arrangement of Gadgets: Reduction from Problem 5 to Problem 4

The reduction from Problem 5 to Problem 4 is relatively straightforward. Since an instance of Problem 5 is made up of multiple instances of Problem 4 with interacting sum-diagonals, it is enough to lay those instances out on the same long diagonal of a larger grid, separated enough that their rows, columns, and difference-diagonals do not have any effect on each other. Apart from combining the separate subproblems correctly, we must ensure that no queens can be wrongly placed in any square off the intended long diagonal. To do this, we exclude all sum-diagonals which do not intersect with any of the instances. Figure 5 gives a high-level illustration of the reduction.

Definition 20 (Reduction from Problem 5 to Problem 4)**.** *Let* $M_5 = \{M_{4,a} \mid a = 0, 1, 2 \ldots k-1\}$ *be an instance of Problem 5 where* $k = |M_5|$ *and each* $M_{4,a} = \langle n_a, C_a, R_a, D^-_a, \{\}\rangle$*. We first set* $n' = \max\{n_a \mid 0 \leq a < k\}$*. We will place each gadget* $M_{4,a}$ *at position* (α_a, β_a) *in a*

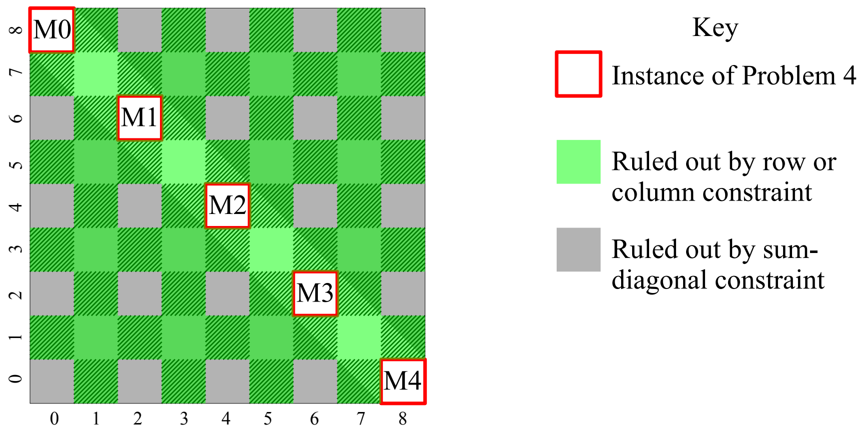


Figure 5: A schematic view of the reduction from Problem 5 to Problem 4 in Definition 20 for an instance of Problem 5 containing 5 instances M0 to M4 of Problem 4. Each square shown represents an $n' \times n'$ cell, where n' is the size of the largest subproblem. The instances M0 to M4 are placed on the long sum-diagonal, from $n'(0, 8)$ for M0 to $n'(8, 0)$ for M4, indicated by red outlines. Green areas indicate that all cells on odd-numbered rows and odd-numbered columns are disallowed. Sum-diagonals that are not on the long diagonal are excluded, and this is indicated by the grey areas. The additional rows, columns and diagonals excluded within each red cell are not shown.

$n \times n$ grid and define the instance Problem 4 as $M_4 = \langle n, C, R, D^-, D^+ \rangle$, where:

$$
\begin{aligned}
\alpha_a &\stackrel{def}{=} 2n'a \\
\beta_a &\stackrel{def}{=} 2n'(k-1-a) \\
n &\stackrel{def}{=} n'(2k-1) \\
C &\stackrel{def}{=} \bigcup_{0 \le a < k} \{c + \alpha_a \mid c \in C_a\} \\
R &\stackrel{def}{=} \bigcup_{0 \le a < k} \{r + \beta_a \mid r \in R_a\} \\
D^- &\stackrel{def}{=} \bigcup_{0 \le a < k} \{d + \alpha_a - \beta_a \mid d \in D_a^-\} \\
D^+ &\stackrel{def}{=} \{0, 1, 2, \ldots 2n'(k-1) - 1\} \cup \{2n'k, 2n'k + 1, \ldots 4n'k - 2\}
\end{aligned}
$$

The definition of D^+ is very different from the others, which are simple aggregations. The role of D^+ is to exclude all diagonals which are outside the range $[2n'(k-1), 2n'k)$. The

reason for this is to exclude queens at positions which cannot be excluded by either C or R but which cannot be allowed: these are seen as grey cells in Figure 5. The well-definedness of M_4 is straightforward, given the well-definedness of each component $M_{4,a}$.

Lemma 21. *If $S_5 = \{S_{4,a} \mid 0 \leq a < k\}$ is a solution to M_5, then $S_4 = \bigcup_{0 \leq a < k}(S_{4,a} + (\alpha_a, \beta_a))$ is a solution to M_4.*

Proof. The queen $\mathbf{q}_a = (\alpha, \beta) \in S_{4,a}$ maps to $\mathbf{q}_{4,a} = (\alpha + \alpha_a, \beta + \beta_a) \in S_4$. First we show no two queens in S_4 attack each other. Two queens in S_4 mapped from the same $S_{4,a}$ both have the same offset (α_a, β_a), so cannot attack each other since no queens in $S_{4,a}$ do. For queens not from the same $S_{4,a}$, first consider columns, rows and difference-diagonals, and queens $\mathbf{q}_a \in S_{4,a}$, $\mathbf{q}_b \in S_{4,b}$, $a \neq b$. Lemma 9 applies with (in terms of that lemma) $n = n'$, $\mathbf{q}_1 = \mathbf{q}_a$, $\mathbf{q}_2 = \mathbf{q}_b$, $q_3 = (2a, 2(k-1-a))$, $q_4 = (2b, 2(k-1-b))$, so $\mathbf{q}_a$ and $\mathbf{q}_b$ can only possibly attack each other along sum-diagonals. Note that the sum-diagonal for $\mathbf{q}_a$ is $\alpha + \beta$, while $\mathbf{q}_{4,a}$ has sum-diagonal $\alpha + \alpha_a + \beta + \beta_a = \alpha + \beta + 2n'a + 2n'(k-1-a) = \alpha + \beta + 2n'(k-1)$. Thus the sum-diagonal in S_4 is the constant $2n'(k-1)$ plus the value of the sum-diagonal in $S_{4,a}$. So two sum-diagonals in S_4 are equal iff the corresponding sum-diagonals are equal in S_5. Since S_5 is a solution, that is not the case, so no two sum-diagonals in S_4 are equal. Hence no two queens in S_4 attack each other.

We must also show that all restrictions based on C, R, D^-, D^+ are obeyed by $\mathbf{q}_{4,a}$. For C note that $\alpha \in C_a$ since S_5 solves M_5, and so $\alpha + \alpha_a \in C$, as required. The case of R is similar. The difference-diagonal of $\mathbf{q}_{4,a}$ is $(\alpha + \alpha_a) - (\beta + \beta_a)$. But $\alpha - \beta \notin D_a^-$ because D_a^- represents the forbidden difference-diagonals of $M_{4,a}$. Therefore $(\alpha + \alpha_a) - (\beta + \beta_a) \notin \{d + \alpha_a - \beta_a \mid d \in D_a^-\}$. We need to check that we cannot have $(\alpha + \alpha_a) - (\beta + \beta_a) \in \{d + \alpha_b - \beta_b \mid d \in D_b^-\}$ for some $b \neq a$, i.e. the difference $(\alpha + \alpha_a) - (\beta + \beta_a) - (d + \alpha_b - \beta_b)$ cannot be zero. Rearranging we have $\alpha - \beta - d + 2n'(a - b - (k-1-a) + (k-1-b))$ $= \alpha - \beta - d + 4n'(a-b)$. $|\alpha - \beta - d| \leq 2n'$ and $|4n'(a-b)| \geq 4n'$, so the sum cannot be zero. Therefore the difference-diagonal of $\mathbf{q}_{4,a}$ is not in D^-. The sum-diagonal of $\mathbf{q}_{4,a}$ is $(\alpha + \alpha_a) + (\beta + \beta_a) = \alpha + 2n'a + (\beta + 2n'(k-1-a)) = \alpha + \beta + 2n'(k-1)$. But $0 \leq \alpha + \beta \leq 2(n'-1)$ so $2n'(k-1) \leq \alpha + \beta + 2n'(k-1) < 2n'k$, so the sum-diagonal of $\mathbf{q}_{4,a}$ is not in D^+. □

Lemma 22. *If S_4 is a solution to M_4, then $S_5 = \{S_{4,a} \mid 0 \leq a < k\}$ where $S_{4,a} = \{(\alpha, \beta) \mid 0 \leq \alpha, \beta < n', ((\alpha, \beta) + (\alpha_a, \beta_a)) \in S_4\}$ is a solution to M_5.*

Proof. The key point is to show that all queens $\mathbf{q} \in S_4$ are of the form $\mathbf{q} = (\alpha + \alpha_a, \beta + \beta_a)$ with $0 \leq \alpha, \beta < n'$, for some a. The construction of C and R gives $\mathbf{q} = (\alpha + \alpha_a, \beta + \beta_b)$ with $0 \leq \alpha, \beta < n'$, but we have to show that $a = b$. The sum-diagonal d of $\mathbf{q}$ is $d = \alpha + \alpha_a + \beta + \beta_b = \alpha + \beta + 2n'a + 2n'(k-1-b) = \alpha + \beta + 2n'(k-1+(a-b))$. But $0 \leq \alpha + \beta \leq 2n' - 2$, so if $a < b$ then $d < 2n'(k-1) - 1$, while if $a > b$ then $d \geq 2n'k$. In either case $d \in D^+$, so we must have $a = b$. All of $\alpha \in C_a$, $\beta \in R_a$, $\alpha - \beta \notin D_a^-$ now follow immediately from the corresponding facts for S_4. We have $\|S_{4,a}\| < n_a$ because each $\max(C_a \cup R_a) < n_a$. Certainly no two queens in any $S_{4,a}$ can attack each other since none in S_4 do. Finally, no two queens in $S_{4,a}$, $S_{4,b}$ can be on the same sum-diagonal since the corresponding queens in S_4 are not and the sum-diagonals in S_4 are the constant value $2n'(k-1)$ greater than the sum-diagonals in each of $S_{4,a}$, $S_{4,b}$. □

Theorem 23. *The reduction of Definition 20 is a parsimonious reduction from Problem 5 to Problem 4.*

Proof. From Lemma 21 and Lemma 22, and the fact that the maps defined in those lemmas are inverses of each other. □

4.4 Gadgets for Clauses and Variables: Reduction from 1-in-3-SAT to Problem 5

For an instance $M_6 = \langle V, C\rangle$ of Restricted 1-in-3-SAT, we will construct an instance $M_5 = \{G_a\}$ of Problem 5. Recall that each $G_a = \langle n_a, C_a, R_a, D_a^-, \{\}\rangle$ is an instance of Problem 4.[5] We assume that $V = \{v_i \mid i = 0, 1, \ldots |V| - 1\}$ and $C = \{c_i \mid i = 0, 1, \ldots |C| - 1\}$. The ordering of both variables and clauses is arbitrary but must be fixed before the reduction is performed. We introduce the notation $G_a \oplus \delta_a$, which can be thought of as shifting G_a by δ_a places along the x-axis.

Definition 24. *For an integer $\delta_a \geq 0$ and $G_a = \langle n_a, C_a, R_a, D_a^-, \{\}\rangle$, we define $G_a \oplus \delta_a \stackrel{def}{=} \langle n_a + \delta_a, \{c + \delta_a \mid c \in C_a\}, R_a, \{d + \delta_a \mid d \in D_a^-\}, \{\}\rangle$.*

Lemma 25. *A set of queens Q_a is a solution to G_a iff $Q_a + (\delta_a, 0)$ is a solution to $G_a \oplus \delta_a$*

Proof. The addition of $(\delta_a, 0)$ does not affect attacks between two queens, so we only have to consider the other properties of $Q_a + (\delta_a, 0)$, and each is easy. Q_a fits on board size n_a iff $Q_a + (\delta_a, 0)$ fits on size $n_a + \|(\delta_a, 0)\| = n_a + \delta_a$. For queen $(\alpha, \beta) \in Q_a$, we have $\alpha \in C_a$ iff $\alpha + \delta_a = \mathrm{col}((\alpha, \beta) + (\delta_a, 0)) \in \{c + \delta_a \mid c \in C_a\}$, and similarly for R_a. Finally, $(\alpha - \beta) \notin D_a^-$ iff $\mathrm{col}((\alpha, \beta) + (\delta_a, 0)) - \mathrm{row}((\alpha, \beta) + (\delta_a, 0)) = \alpha - \beta + \delta_a \notin \{d + \delta_a \mid d \in D_a^-\}$. □

G^0: A gadget for a SAT variable. A copy of the same gadget G^0 is used for each SAT variable in a problem. Each copy of G^0 has two solutions, and we will arrange that one solution will indicate that the corresponding SAT variable should be true, while the other solution will indicate that the SAT variable should be false. The gadget G^0 is illustrated in Figure 6 and defined in Definition 26. G^0 has only two solutions, as can be checked by a very small search. The two solutions are $\{(0, 0), (1, 3), (3, 2)\}$, which we call the 'red solution', and $\{(0, 2), (1, 0), (3, 3)\}$, the 'blue solution'. These use disjoint sets of sum-diagonals. These are respectively $\{0, 4, 5\}$ (the 'red diagonals') and $\{1, 2, 6\}$ (the 'blue diagonals'). The gadget G^0 was found by a search using Savile Row (Nightingale et al., 2017) and Minion (Gent et al., 2006) from a model available online.[6]

For a variable v_i in a clause c we will use one red diagonal and one blue diagonal from the i^{th} copy of G^0. We will write $d_{i,c}^r$ for the red diagonal being used, and $d_{i,c}^b$ for the blue diagonal, and $d_{i,c}^x$ for a diagonal of indeterminate colour (i.e. $x \in \{b, r\}$). Note that in Restricted 1-in-3-SAT, v_i occurs in at most three clauses, and no more than once in any clause. Because of this, for any given variable v, the value $d_{v,c}^x$ determines the values c and x.

5. We use the letter G instead of M_4 for gadgets in M_5 to avoid proliferation of subscripts and to emphasise that these are gadgets, which will be used to encode different parts of a 1-in-3-SAT instance.
6. `https://ipg.host.cs.st-andrews.ac.uk/n-queens-completion-experiments-code.tgz`

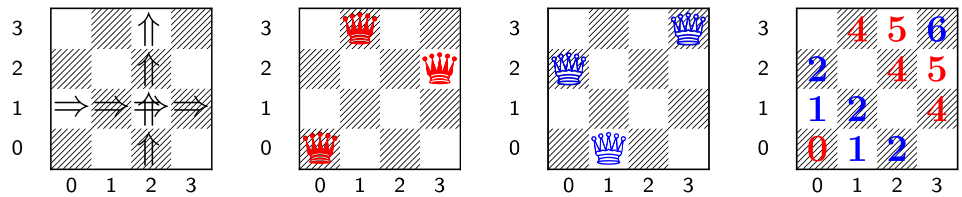


Figure 6: Illustration of the 4×4 gadget G^0. The first diagram shows a board with arrows barring the second row and third column. The second and third boards show the only two ways that three queens can be placed in the remaining squares with no two queens attacking each other. The first, red, solution will be used to indicate that the corresponding SAT variable is true, with the second, blue, solution for false. The final board shows that the red and blue solutions occupy disjoint sets of sum-diagonals. Each square is labelled with the sum-diagonal it is on, with diagonals 0, 4, and 5 being red and 1, 2, and 6 being blue. The other sum-diagonal (3) is not used in either solution.

Definition 26 (Gadget G^0)**.** *We define* $G^0 \stackrel{def}{=} \langle 4, \{0,1,3\}, \{0,2,3\}, \{\}, \{\}\rangle$. *For a variable* v_i *where* $0 \leq i < |V|$, *we define* $\delta_{v_i} \stackrel{def}{=} 21i$. *(We sometimes write* δ_i *for* δ_{v_i} *to avoid the double subscript.) Each variable* v_i *is associated with one copy of* G^0 *named* G^0_i, *which is defined as follows:* $G^0_i \stackrel{def}{=} G^0 \oplus \delta_{v_i}$.

We allocate red and blue diagonals as follows. If v_i *occurs in clauses* $c_1 < c_2 < c_3$, *then for* c_1 *we set* $d^r_{i,c_1} \stackrel{def}{=} 0, d^b_{i,c_1} \stackrel{def}{=} 1$; *for* c_2 *we set* $d^r_{i,c_2} \stackrel{def}{=} 4, d^b_{i,c_2} \stackrel{def}{=} 2$; *and for* c_3 *we set* $d^r_{i,c_3} \stackrel{def}{=} 5, d^b_{i,c_3} \stackrel{def}{=} 6$. *If* v_i *occurs in fewer than three clauses we simply define the relevant subset of the* $d^x_{i,c}$.

G^1, G^2, G^3: Three gadgets for a 1-in-3-SAT clause. Now we present the gadgets associated with a clause $c = \{v_i, v_j, v_k\}$, namely G^1_c, G^2_c, G^3_c. G^1_c will ensure that at least one of the G^0_i, G^0_j and G^0_k takes its red solution, while G^2_c and G^3_c will each ensure that at least one of them (different in each case) takes its blue solution. Combined they will force *exactly* one of G^0_i, G^0_j and G^0_k to take its red solution: thus variables assigned true in SAT will correspond precisely with G^0 gadgets using their red solutions. The gadgets G^1_c, G^2_c and G^3_c will vary from clause to clause, unlike G^0, where the gadgets are the same for each variable. For a variable $v \in \{v_i, v_j, v_k\}$ and colour $x \in \{b, r\}$ we define $col^x_v \stackrel{def}{=} \delta_v + d^x_{v,c}$: all columns in G^1_c, G^2_c, G^3_c will be of this form. The rows in G^1_c, G^2_c, G^3_c will also be of standard form, drawn from $\{7a \mid a \in 0,1,2\} = \{0,7,14\}$.

Definition 27 (Gadget G^1_c)**.** *Suppose we have a clause* $c = \{v_i, v_j, v_k\}$ *of Restricted 1-in-3-SAT where* $i < j < k$, *so also* $\delta_i < \delta_j < \delta_k$.

- $G^1_c \stackrel{def}{=} \langle col^b_k + 1, C_1, R_1, D^-_1, \{\}\rangle$ *where:* $C_1 = \{col^b_i, col^b_j, col^b_k\}$, $R_1 = \{0, 7, 14\}$, *and* $D^-_1 = \{col^b_k - 7,\ col^b_i - 14\}$.

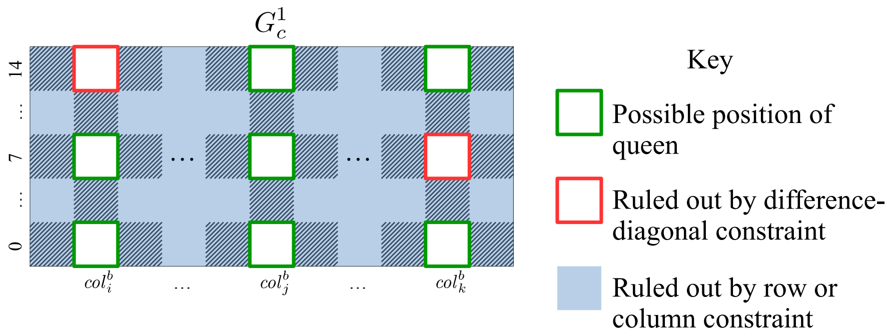


Figure 7: Illustration of G_c^1 showing the seven positions where queens may be placed.

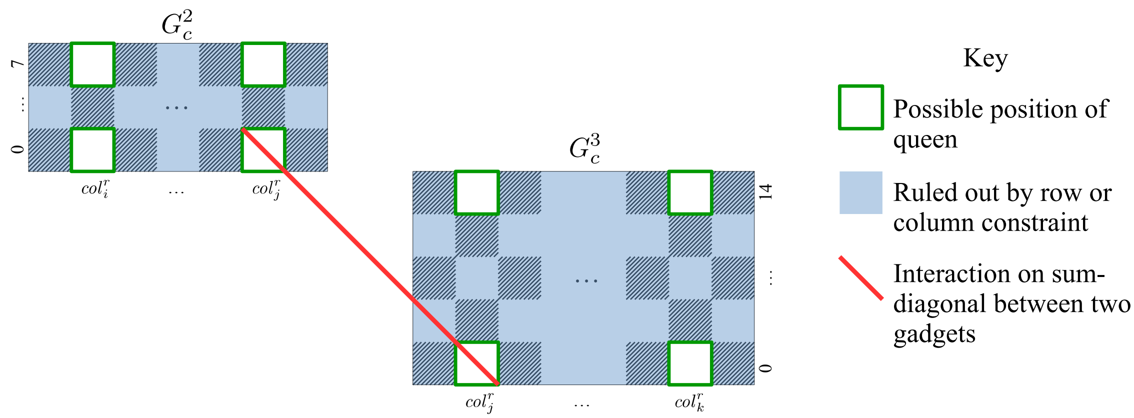


Figure 8: Illustration of G_c^2 and G_c^3 and their interaction through one sum-diagonal (col_j^r+0).

Gadget G_c^1 is illustrated in Figure 7. There are three solutions of G_c^1, and each solution places a queen in row 0 at a different column. G_c^1 interacts with the three relevant G^0 gadgets through the position of the queen on row 0. For example, if the queen is placed at $(col_j^b, 0)$ then the G^0 gadget for variable v_j is forced to take its red solution. G_c^1 does not directly interact with any other gadget.

Definition 28 (Gadgets G_c^2 and G_c^3). *Suppose $c = \{v_i, v_j, v_k\}$ where $i < j < k$, so $\delta_i < \delta_j < \delta_k$.*

- $G_c^2 \overset{def}{=} \langle col_j^r + 1, C_2, R_2, \{\}, \{\}\rangle$ *where:* $C_2 = \{col_i^r, col_j^r\}$, *and* $R_2 = \{0, 7\}$.
- $G_c^3 \overset{def}{=} \langle col_k^r + 1, C_3, R_3, \{\}, \{\}\rangle$ *where:* $C_3 = \{col_j^r, col_k^r\}$, *and* $R_3 = \{0, 14\}$.

Gadgets G_c^2 and G_c^3 are illustrated in Figure 8. Both these gadgets have two solutions. They both interact with G^0 gadgets via the queen placed on row 0. They each force one

of the G^0 gadgets to take its blue solution. G_c^2 and G_c^3 cannot both use the specific pair $(col_j^r, 0)$ because both have the same sum-diagonal. Taken together, G_c^2 and G_c^3 have three solutions, each of which forces two G^0 gadgets to take their blue solutions.

It is natural to ask whether we could encode 3-SAT clauses using only G_c^1 (adapted to use blue diagonals for the positive literals and red diagonals for negative literals), making G_c^2 and G_c^3 redundant. 3-SAT with at most 3 occurrences of each variable is NP-complete. However, a 3-SAT clause c is allowed to be satisfied by more than one of its literals. In this case the corresponding G_c^1 gadget would have more than one solution, so the reduction would not be parsimonious. Supposing we wished to prove only NP-Completeness we could use the simpler reduction from 3-SAT. However, we know of no way to simplify the other three reductions in Definitions 10, 16, and 20.

The sum-diagonals for row 0 in G_c^1 are $\delta_i + d_{i,c}^b$, $\delta_j + d_{j,c}^b$, $\delta_k + d_{k,c}^b$, in each case a blue sum-diagonal from the relevant G^0 gadget. Similarly, the sum-diagonals for row 0 in G_c^2, G_c^3 are $\delta_i + d_{i,c}^r$, $\delta_j + d_{j,c}^r$, $\delta_k + d_{k,c}^r$, each a red sum-diagonal from the relevant G^0.

The separations of 7 between rows and 21 between columns were chosen to make the following simple arithmetical fact true:

Observation 29. *Suppose that we have integers $a_1, a_2, c_1, c_2, d_1, d_2$ with $0 \leq a_1, a_2 \leq 2$, $0 \leq d_1, d_2 \leq 6$, and suppose that $21(c_1 - c_2) \pm 7(a_1 - a2) \pm (d_1 - d_2) = 0$. Then $c_1 = c_2$, $a_1 = a_2$ and $d_1 = d_2$.*

Proof. If $c_1 \neq c_2$ then $|21(c_1 - c_2)| \geq 21$, while $|\pm 7(a_1 - a_2) \pm (d_1 - d2)| \leq (2-0)7 + 6 = 20 < 21$. This makes it impossible to make the sum equal to zero unless $c_1 = c_2$, and also $7(a_1 - a_2) \pm (d_1 - d2) = 0$. By similar reasoning, we must have $a_1 = a_2$ since if $a_1 \neq a_2$ then $|7a_1 - 7a_2| \geq 7$, and $|d_1 - d_2| \leq 6 < 7$. Since we have $c_1 = c_2$, $a_1 = a_2$, we must also have $d_1 = d_2$. □

Before presenting the reduction itself, we prove (for all three gadget types) that the possible locations of the queens do not attack each other diagonally.

Lemma 30. *Within each of G_c^1, G_c^2, G_c^3, where the gadget has columns C_i and rows R_i, the difference-diagonals $\alpha - \beta$ with $\alpha \in C_i, \beta \in R_i$ are all distinct. The same applies for the sum-diagonals $\alpha + \beta$.*

Proof. We can deal with both cases at once by writing $\pm$ where necessary. For $v_1, v_2 \in \{i, j, k\}$, $x \in \{b, r\}$, $a_1, a_2 \in \{0, 1, 2\}$ we have $\alpha = col_{v_1}^x$, $\beta = 7a_1$. This makes $\alpha \pm \beta = \delta_{v_1} + d_{v_1,c}^x \pm 7a_1$. If two sum-/difference-diagonals within one G_c^1, G_c^2, or G_c^3 are the same then we have (since values of x and c are fixed within a clause): $0 = 21(v_1 - v_2) \pm 7(a_1 - a_2) + d_{v_1,c}^x - d_{v_2,c}^x$. Observation 29 gives $v_1 = v_2$ and $a_1 = a_2$. In each gadget G_c^1, G_c^2, G_c^3 the value of v and a uniquely determines α and β. Thus the value of $\alpha \pm \beta$ uniquely determines α and β, which can only happen if all sum-/difference-diagonals are distinct. □

Definition 31 (Reduction from Restricted 1-in-3-SAT to Problem 5)**.** *Consider an instance $M_6 = \langle V, C \rangle$ of Restricted 1-in-3-SAT. For a clause $c = \{v_i, v_j, v_k\} \in C$ we will write $M_{5,c}$ for the instance of Problem 5, $M_{5,c} \stackrel{def}{=} \{G_i^0, G_j^0, G_k^0, G_c^1, G_c^2, G_c^3\}$. For the reduction of the entire instance M_6 we set:*

$$M_5 \stackrel{def}{=} \bigcup_{c \in C} M_{5,c}$$

Each G_v^0 can occur in the union up to three times, if v is in three clauses. However, each copy is identical since G_v^0 does not depend on c, so in the final set M_6, each G_v^0 occurs exactly once.

Lemma 32. *The possible sum-diagonals in M_5 are all of the form $\delta_v + d_{v,c}^x + 7a$ with $0 \le a \le 2$, and additionally: $a = 0$ in G_v^0; $x = b$ in G_c^1; and $x = r$ in G_c^2, G_c^3. Furthermore, if $\delta_{v_1} + d_{v_1,c_1}^{x_1} + 7a_1 = \delta_{v_2} + d_{v_2,c_2}^{x_2} + 7a_2$, then we have all of $v_1 = v_2$, $c_1 = c_2$, $x_1 = x_2$, $a_1 = a_2$.*

Proof. The first statement follows from a straightforward case analysis. In G_v^0 the possible values are exactly $\delta_v + d_{v,c}^x$. In each of G_c^1, G_c^2, G_c^3 we have $C \subseteq \{col_i^x, col_j^x, col_k^x\}$, $R \subseteq \{7a \mid 0 \le a \le 2\}$, and so the sum-diagonal is $col_v^x + 7a = \delta_v + d_{v,c}^x + 7a$. In G_c^1, $x = b$ and in G_c^2, G_c^3, $x = r$, as required by the statement. For the second statement, suppose $21(v_1 - v_2) + 7(a_1 - a_2) + (d_{v_1,c_1}^{x_1} - d_{v_2,c_2}^{x_2}) = 0$. Observation 29 applies to immediately give $v_1 = v_2$, $a_1 = a_2$, and also $d_{v_1,c_1}^{x_1} = d_{v_2,c_2}^{x_2} = d_{v_1,c_2}^{x_2}$. By construction of G^0, for a given variable v_1, the values c and x are uniquely determined from the value of $d_{v_1,c}^x$. Hence $c_1 = c_2$ and $x_1 = x_2$. □

Lemma 33. *For a clause $c = \{v_i, v_j, v_k\}$ from M_6, $M_{5,c}$ has exactly three solutions. In one, G_i^0 takes its red solution and G_j^0, G_k^0 take their blue solutions; in a second G_j^0 is red and G_i^0, G_k^0 blue; and in the third G_k^0 is red and G_i^0, G_j^0 blue.*

Proof. We work by case analysis on where the queen in row 0 in G_c^1 is placed. This can be one of the pairs $(col_i^b, 0)$, $(col_j^b, 0)$, or $(col_k^b, 0)$. In each case we will construct a solution set of queens $S_4(G)$ for each $G \in M_{5,c}$, and show that $S_{5,c} = \{S_4(G) \mid G \in M_{5,c}\}$ is the unique solution to $M_{5,c}$ for that case. In the case for $(col_x^b, 0)$, the corresponding gadget G_x^0 will use its red solution and the other two G^0 their blue solutions.

First consider the case $(col_i^b, 0) \in S_4(G_c^1)$. This has the sum-diagonal $col_i^b = d_{i,c}^b + \delta_i$, the same value as a blue sum-diagonal of G_i^0, so we set $S_4(G_i^0)$ to its red solution. The other two pairs in $S_4(G_c^1)$ must use the columns col_j^b and col_k^b. However, the pair $(col_k^b, 7)$ is excluded by D_1^-. Hence we set $S_4(G_c^1) = \{(col_i^b, 0), (col_j^b, 7), (col_k^b, 14)\}$. Now consider G_c^2. The columns available in row 0 are col_i^r and col_j^r. But the pair $(col_i^r, 0)$ has the sum-diagonal $col_i^r = d_{i,c}^r + \delta_i$, and this is the same value as one red sum-diagonal of G_i^0, and we are using the red solution for G_i^0 so we cannot use the pair $(col_i^r, 0)$ in G_c^2. So we set $S_4(G_c^2) = \{(col_j^r, 0), (col_i^r, 7)\}$. But the sum-diagonal value of $(col_j^r, 0)$ is the same as a red diagonal in G_j^0, so we set $S_4(G_j^0)$ to its blue solution. Now consider G_c^3. The pair $(col_j^r, 0)$ is disallowed because it is on the same sum-diagonal as the pair $(col_j^r, 0) \in S_4(G_c^2)$. Hence we set $S_4(G_c^3) = \{(col_k^r, 0), (col_j^r, 14)\}$. The sum-diagonal value of $(col_k^r, 0)$ forces us to set $S_4(G_k^0)$ to its blue solution.

The above has defined $S_{5,c}$ completely for this case, and shown that there were no alternative choices at any point. This establishes uniqueness, but it is still necessary to show that $S_{5,c}$ is indeed a solution to $M_{5,c}$, i.e. that each $S_4(G)$ is a solution to G and that there are no sum-diagonal clashes between gadgets. The values $S_4(G_v^0)$ are already known to be solutions. Each $S_4(G^1)$, $S_4(G^2)$, $S_4(G^3)$ places exactly one queen in each allowed row

and column. Lemma 30 shows that no two queens can be placed on the same difference- or sum-diagonal within G_c^1, G_c^2, G_c^3. Also $S_4(G_c^1)$ correctly does not contain the pairs $(col_k^b, 7)$ and $(col_i^b, 14)$ which are disallowed by D_1^-. We now consider sum-diagonals from different gadgets. From Lemma 32, if two sum-diagonals have the same value of $\delta_v + d_{v,c}^x + 7a$, they have the same values of v, c, x, a. We only have to consider $a = 0$, since from Lemma 32, G^0 only has $a = 0$, G_c^2 and G_c^3 do not share a non-zero value of a, and G_c^1 does not share a value of x with G_c^2, G_c^3. For $a = 0$ the only cases of duplicate values of $\delta_v + d_{v,c}^x$ are: between G_v^0 and G_c^1, G_c^2, G_c^3; and the value $\delta_j + d_{j,c}^r$ between G_c^2 and G_c^3. The construction of $S_{5,c}$ above avoided using any of these sum-diagonals twice, and thus is indeed a solution.

We omit details of the cases $(col_j^b, 0)$ and $(col_k^b, 0)$ in G_c^1, which force G_j^0 and G_k^0 (respectively) to take their red solutions with the other copies of G^0 blue. The case $(col_k^b, 0)$ is exactly similar to the case above except that D_1^- rules out the pair $(col_i^b, 14)$. There is a slight difference for the pair $(col_j^b, 0)$ in G_c^1: the solution of G_c^1 is fixed by the exclusion of both diagonals in D_1^-, while the solutions of G_c^2 and G_c^3 are unique because $(col_j^r, 0)$ cannot be used in either since it has the same sum-diagonal as the red solution to G_j^0. □

Lemma 34. *For any truth assignment S_6 which solves M_6 then S_5 is a solution to M_5, where $S_5 = \{S_4(G) \mid G \in M_5\}$; and where $S_4(G_i^0)$ is the red solution iff $S_6(v_i) = true$ and the blue solution otherwise; and where values of $S_4(G^1)$, $S_4(G^2)$, $S_4(G^3)$ are set as in the proof of Lemma 33.*

Proof. Because S_6 is a solution to $M_6 = \langle V, C\rangle$, for any clause $c = \{v_i, v_j, v_k\} \in C$ we know that S_6 assigns exactly one of the literals in c to true. Hence exactly one of $S_4(G_i^0)$ is red and the other two are blue, and, by Lemma 33, the values of $S_4(G_c^1)$, $S_4(G_c^2)$, $S_4(G_c^3)$ are well-defined solutions and unique. It remains to show that for any two gadgets $G, G' \in M_5$, and any two queens $\mathbf{q} \neq \mathbf{q}'$, $\mathbf{q} \in S_4(G)$, $\mathbf{q}' \in S_4(G')$, the sum-diagonals of $\mathbf{q}$ and $\mathbf{q}'$ are different. Lemma 32 shows that if $\mathbf{q}$ and $\mathbf{q}'$ have the same sum-diagonals, then they have the same values (in terms of that lemma) of v_1 and v_2, and of c_1 and c_2, that is they arise from the same combination of clause and variable. Hence either $G = G' = G_v^0$ or $G, G' \in M_{5,c}$, or both. In either case, their sum-diagonals must be different according to Lemma 33. Finally we note that S_5 is well defined and unique because each $S_4(G_i^0)$ is forced to be either the red or blue solution by the truth value of v_i, even if v_i appears in no clauses. □

Lemma 35. *For any solution $S_5 = \{S_4(G) \mid G \in M_5\}$ to M_5, then S_6 is a solution to M_6, where $S_6(v_i) = true$ iff $S_4(G_i^0)$ is the red solution and $S_6(v_i) = false$ iff $S_4(G_i^0)$ is the blue solution.*

Proof. S_6 is well defined since each G_i^0 can only take its red or blue solution. From Lemma 33, exactly one variable from each clause in M_6 is set to true by S_6. □

Theorem 36. *The reduction of Definition 31 from Restricted 1-in-3-SAT to Problem 5 is a parsimonious reduction.*

Proof. From Lemma 34 and Lemma 35, and the fact that the maps defined in those lemmas are inverses of each other. □

Theorem 37. *The n-Queens Completion problem (Problem 2) is NP-Complete and #P-Complete.*

Proof. Theorems 36, 23, 19, and 13 give successive parsimonious reductions starting from Restricted 1-in-3-SAT and ending with n-Queens Completion. By inspection each reduction can be computed in polynomial time. Restricted 1-in-3-SAT is NP-Complete and #P-Complete by Theorem 4 so this gives NP- and #P-hardness. NP- and #P-easiness follow since potential solutions to n-Queens Completion are straightforward to check. □

As an intermediate problem, the Excluded Diagonals Problem (Problem 4) is also NP- and #P-Complete. As a straightforward corollary we obtain the NP-Completeness of the "Blocked n-Queens" problem, which has been used as a benchmark for answer set programming (ASP). In this problem, some squares are blocked from holding queens (Namasivayam & Truszczynski, 2009).

Problem 7 (Blocked n-Queens). Problem*:* $M_7 = \langle n, P\rangle$ *where* n *is an integer and* P *is a set of queens (possibly attacking each other) that all fit on board size* n*.* Solution*: A set* S_7 *of queens which is a solution to the* n*-Queens Problem for* n *and such that* $P \cap S_7 = \{\}$*.*

Corollary 38. *The Blocked* n*-Queens problem is NP-Complete and #P-Complete.*

Proof. The reduction from n-Queens Completion to Blocked n-Queens is straightforward. For a problem $M_2 = \langle n, P_2\rangle$ we block each queen on the same row as one of the queens in P_2. That is, we form the set $P_7 = \bigcup_{(\alpha,\beta)\in P_2}\{(i,\beta) \mid 0 \le i < n, i \ne \alpha\}$. Any solution to M_7 $= \langle n, P_7\rangle$ necessarily contains each queen in P_2 so is a solution to M_2. This establishes NP- and #P-hardness, while -easiness is true because potential solutions to Blocked n-Queens problems can be checked easily. □

5. Empirical Study

We have shown that n-Queens Completion and Blocked n-Queens are NP-Complete, but how hard are they to solve in practice? This is an important question given the repeated controversy surrounding the use of n-Queens as a benchmark problem discussed in Section 1. We first describe solvers based on three different technologies: a special purpose solver; a SAT solver; and a constraint solver. We describe methods of generating random benchmark instances of three problem classes: Blocked n-Queens, n-Queens Completion, and the Excluded Diagonals Problem (Problem 4). We looked for *phase transitions* (Hogg, Huberman, & Williams, 1996) between solvable and unsolvable instances, in common with other NP-Complete problems, for example Boolean satisfiability (SAT) (Mitchell, Selman, & Levesque, 1992; Achlioptas, 2009; Altarelli, Monasson, Semerjian, & Zamponi, 2009). It is common in NP-Complete problems to see that randomly generated instances are hardest at the transition from solvable to unsolvable instances (Cheeseman, Kanefsky, & Taylor, 1991; Mitchell et al., 1992; Gent, MacIntyre, Prosser, & Walsh, 1995). We do see this pattern for every solver in two of our generators but not in the third: a straightforward generation method for n-Queens Completion appears unable to produce instances that are hard to solve using SAT.

Experiments were performed on a 32-core AMD Opteron 6272 at 2.1 GHz with 256 GB RAM. Savile Row was executed in OpenJDK Java 1.7.0_131 with the 64-bit server VM. For random instances, we solved 1000 randomly generated instances per data point, and solvers were allowed to run to completion. We have made available our code including the adapted

version of Savile Row that we used,[7] detailed results including some additional graphs,[8] and experimental instances.[9]

5.1 Three Solvers

We constructed three types of solver, each of which is complete, i.e. is guaranteed to either find a solution or confirm that there are none. We make no claim as to how good these methods are compared to other possible methods: we wrote solvers using three different technologies to ensure that if we found hard instances, they were hard for at least a variety of solvers.

The first approach is a special-purpose solver written in C++. It is based on bit manipulation techniques previously used for counting solutions of n-Queens (Richards, 1997). It searches exceptionally fast per search state although the intelligence of the search process is limited. The algorithm is a backtrack search that constructs a solution starting at row 0 and moving up the board. It uses three bit patterns to represent attacks on the current row β via the sum-diagonals (ld_β), columns ($cols_\beta$) and difference-diagonals (rd_β). Each bit pattern contains a 1 in position α iff a queen attacks position (α, β) via the respective diagonal or column. When a queen (α, β) is placed on the board, bit α is set in ld_β, $cols_\beta$, and rd_β. When moving from row β to $\beta+1$, $ld_{\beta+1} = ld_\beta << 1$ (i.e. ld is shifted one position left), $rd_{\beta+1} = rd_\beta >> 1$ (rd is shifted one position right) and $cols_{\beta+1} = cols_\beta$. The attacked positions of row $\beta+1$ are given by the bitwise OR of $ld_{\beta+1}$, $cols_{\beta+1}$, and $rd_{\beta+1}$. Another bit pattern technique is used to find the rightmost non-attacked position that has not yet been tried. We made simple adaptations to account for the fact that some queens are already placed in n-Queens Completion. First, we skip rows that contain a queen. Second, ld_0, $cols_0$, and rd_0 are populated with attacks of the pre-placed queens. Third, when shifting ld (rd) the least (most) significant bit is populated from an array to allow for a pre-placed queen attacking diagonally from above. A different adaptation was made for Blocked n-Queens where the blocked positions of a row β are masked out using a bitwise AND before attempting to place a queen on row β. A third adaptation was made to solve the Excluded Diagonals Problem, in which ld_0 and rd_0 are pre-populated with excluded diagonals, and when shifting ld (rd) the least (most) significant bit is populated from an array containing the excluded diagonals that do not intersect row 0.

The second approach is to encode to SAT and apply a CDCL SAT solver. We use the SAT solver Lingeling (Järvisalo, Heule, & Biere, 2012) which was winner of the Sequential, Application SAT+UNSAT track of the SAT competition 2014.[10] We wrote two Python programs to encode n-Queens Completion and Blocked n-Queens into SAT. SAT variables are added for each square which is not attacked by a preplaced queen (for n-Queens Completion) or not blocked (for Blocked n-Queens). We use the commander-variable encoding (Klieber & Kwon, 2007) for Boolean sum ≤ 1 constraints (using a group size of 3) to add 'exactly-one' constraints for each row and column, and 'at-most-one' constraints for each diagonal. To solve the Excluded Diagonals Problem we implemented the reduction to n-Queens Completion (Definition 10 and Definition 16) with a separate Python program. While this approach

7. `https://ipg.host.cs.st-andrews.ac.uk/n-queens-completion-experiments-code.tgz`
8. `https://ipg.host.cs.st-andrews.ac.uk/n-queens-completion-experiments-results.tgz`
9. `https://ipg.host.cs.st-andrews.ac.uk/n-queens-completion-experiments-instances.tgz`
10. Lingeling version ayv 86bf266b9332599f1b876e28a02fe8427aeaa2db

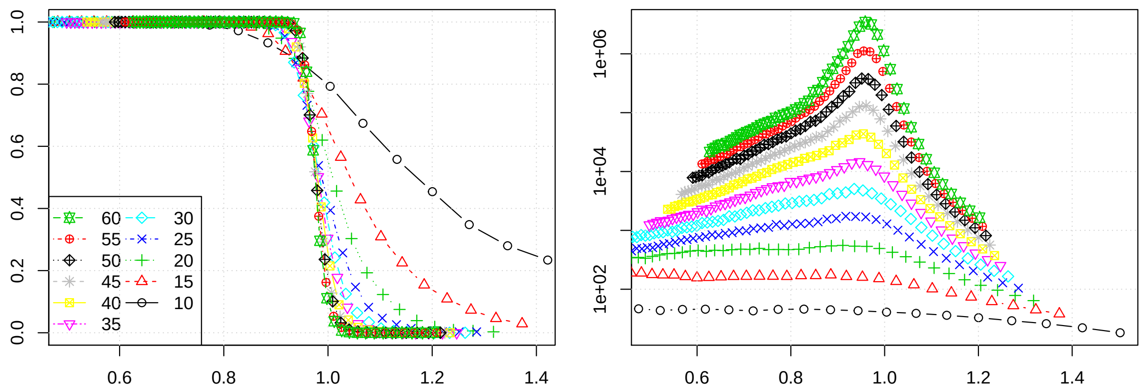


Figure 9: Random Blocked n-Queens instances: y axis shows probability of a solution existing (left) and mean nodes searched by Lingeling (right) against $\kappa_{n,b}$ on x axis. The colour and plot symbol indicate the value of n.

generates large n-Queens Completion instances, the number of non-attacked squares (and therefore the SAT encoding) is small. Further, since we can generate hard instances of the Excluded Diagonals Problem, this approach gives us confidence that we can also generate hard instances of n-Queens Completion.

The third approach uses the constraint solver Minion 1.8 (Gent et al., 2006), with a constraint model optimised by Savile Row 1.6.4 (Nightingale et al., 2017). For each row β, the model has three variables representing the column, sum- and difference-diagonal of the queen placed on row β. The model exploits powerful constraint propagation by using three all-different constraints to enforce non-attacks on columns, sum- and difference-diagonals. Two table constraints are used on each row to connect the column variable to the other two variables. Savile Row preprocesses each instance with Minion which may assign variables. Then it applies partial evaluation to simplify the constraints before generating the Minion input file. n-Queens Completion is implemented by assigning the appropriate column variables, Blocked n-Queens by removing values of the column variables, and Excluded Diagonals Problem by removing values of the variables representing diagonals.

5.2 Blocked n-Queens

We start with Blocked n-Queens (Problem 7) since it has previously been used as a benchmark for ASP solvers (Namasivayam & Truszczynski, 2009; Gebser, Liu, Namasivayam, Neumann, Schaub, & Truszczyński, 2007). We proved the problem NP-Complete and #P-Complete in Corollary 38. The competition instances came from a set of 100 for each $n \in \{28, 48, 50, 56\}$.[11] The SAT solver was able to solve all instances within 8 minutes CPU time, the hardest taking 431.4s total time and 2,890,435 nodes.

To investigate further we designed our own generator, since the generation method for the competition instances is not published. A row or column with zero unblocked squares

11. Thanks to Martin Gebser for providing the 400 instances.

makes an instance trivially unsolvable. Nine of the ASP competition instances (5 at $n = 56$) are trivial for this reason. More subtle problems remain with up to 3 unblocked squares. For example, if the only three unblocked squares in row 1 were in columns 5, 6, and 7, then row 2 could not have a queen in column 6. This is an implicit unary constraint which might combine with others to make a problem trivial: similar issues with 'flawed' instances have been seen in generation methods for several other combinatorial problems (Achlioptas, Molloy, Kirousis, Stamatiou, Kranakis, & Krizanc, 2001; Gent & Walsh, 1999; Gent, Nightingale, Rowley, & Stergiou, 2008). To avoid this danger, we require each row and column to have at least 4 free squares. We generate an instance with b blocked squares as follows. We start with an empty set of blocked squares. We select a non-blocked square uniformly at random, rejecting any that leave a row or column with 3 or fewer unblocked squares, and add it to the set. Squares are added to the set until we have b blocked squares, or all remaining non-blocked squares are rejected (in which case we restart with an empty set).

We calculated a constrainedness parameter κ (Gent, MacIntyre, Prosser, & Walsh, 1996). The expectation is that there should be a phase transition at a critical value of $\kappa \leq 1$, with search hardness peaking near that value. For a random ensemble with state space S and probability p_s that a given state $s \in S$ is a solution, $\kappa = -\log_2 p_s / \log_2 |S|$. We take S to be the set of solutions to the n-Queens problem. It is conjectured that $|S| \sim n!/2.54^n$ (Cloitre, 2002): by Stirling's formula, $|S| \approx \sqrt{2\pi n}(n^n)/(2.54e)^n$. With b blocked squares, each queen in $s \in S$ has probability b/n^2 of being disallowed, so $p_s \approx (1 - b/n^2)^n$. Putting the approximations together and simplifying, we get:

$$\kappa_{n,b} \stackrel{def}{=} \frac{n(\log_2 n - \log_2(n - b/n))}{n(\log_2 n - \log_2 2.54e) + \frac{1}{2}(1 + \log_2 \pi + \log_2 n)} \tag{1}$$

Figure 9 shows experimental results using the SAT solver. We see a transition in solvability at a critical value of $\kappa \approx 0.97$, with the transition sharpening as n increases. There is a hardness peak associated with the solvability transition in each solver. We only show results for the SAT solver as it is much better than the other two, being able to solve instances up to $n = 60$ in reasonable time (peak mean time 681.9s, peak mean nodes $\approx 3.5 \times 10^6$). The Minion-based solver solves only up to 40 (peak mean time 358.9s, peak mean nodes $\approx 1.8 \times 10^6$), and the special purpose solver up to 35 (peak mean time 126.7s, peak mean nodes $\approx 2.6 \times 10^9$). Increasing n by 5 above these values made the experiments prohibitively expensive to run with expected peak mean CPU times above 1000s in each case. Examining Equation (1), the only influence that b has is in the term $n - b/n$, which is a natural parameter, i.e. the mean number of unblocked squares in each row. Given the observed critical value $\kappa \approx 0.97$, solving for $n - b/n$ shows that as n increases the critical value of $n - b/n$ grows very slowly, $\sim (2.54e)^{0.97} n^{0.03} \approx 6.5 n^{0.03}$. Figure 10 shows our experimental results for SAT plotted against $n - b/n$ instead of $\kappa_{n,b}$. Results are consistent with this analysis but are over too limited a range to comment on the scaling.

5.3 Random Placement of Queens

The most natural way to generate random instances of n-Queens Completion is to place queens at random positions. Given n and a number p of pre-placed queens, we start with

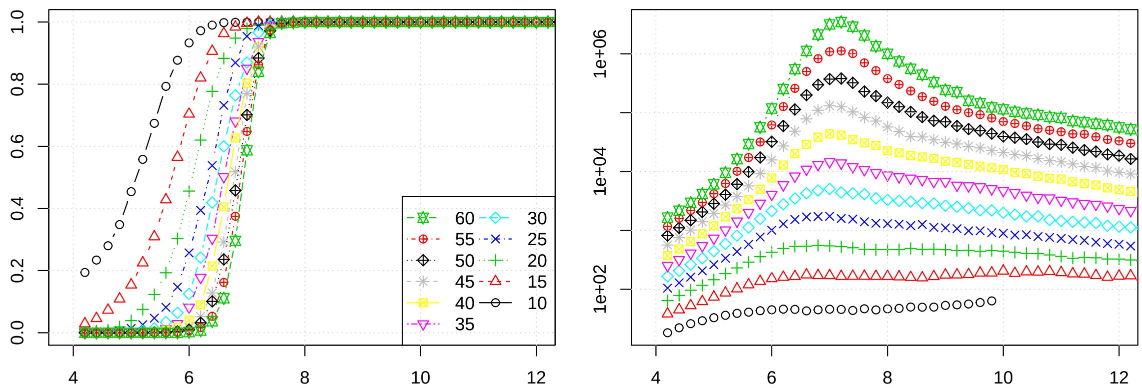


Figure 10: Random Blocked n-Queens instances: y axis gives probability of a solution existing (left) and mean nodes searched by Lingeling (right) against $n - b/n$ (mean number of unblocked squares per row) on x axis. The colour and plot symbol indicate the value of n.

an empty list $P = \{\}$ of pre-placed queens, adding queens iteratively until $|P| = p$. At each step a queen $\mathbf{q}$ is chosen uniformly at random from the set of queens that do not attack any queen already in P. If we allowed queens in P to attack each other then the birthday paradox would mean that we could only generate instances with $p = O(\sqrt{n})$ before almost all instances contained attacks: this would be a flawed generator. We typically experimented with fixing the difference $n - p$, the number of queens to be placed. We do not show detailed results here but graphs and detailed results are included in our online archive. Results with the special purpose solver were encouraging. We saw a transition between solvable and unsolvable regions, associated with hard instances. The hardest instances can require an astronomical number of search nodes: one instance at $n - p = 30, n = 108$ required $\approx 6.64 \times 10^{12}$ nodes, search taking ≈ 1.8 days despite ≈ 42 million nodes being searched per second in a single thread. There was also an apparent hard region with very small values of p, similar to clusters of difficult instances in the mainly satisfiable region in other problems (Hogg & Williams, 1994; Gent & Walsh, 1994). However, when using the SAT solver there are very few difficult instances. Brief investigation of much larger instances (e.g. $n - p = 500$) failed to find any the SAT solver could not solve in a short time and even instances with n in the millions had solutions. While not definitive, these results suggest that this generation method does not lead to instances that are intrinsically hard, and it may not have a phase transition at all for large n. Therefore we conclude it is not a suitable generator for n-Queens Completion benchmarks.

5.4 The Excluded Diagonals Problem

While the simple and natural generator for random n-Queens Completion seems to be unable to generate hard instances, we found that the following generation method for Problem 4 (the Excluded Diagonals Problem) generates consistently hard instances for the three solver types we consider. Given n, we allow all rows and columns but select d different diagonals to

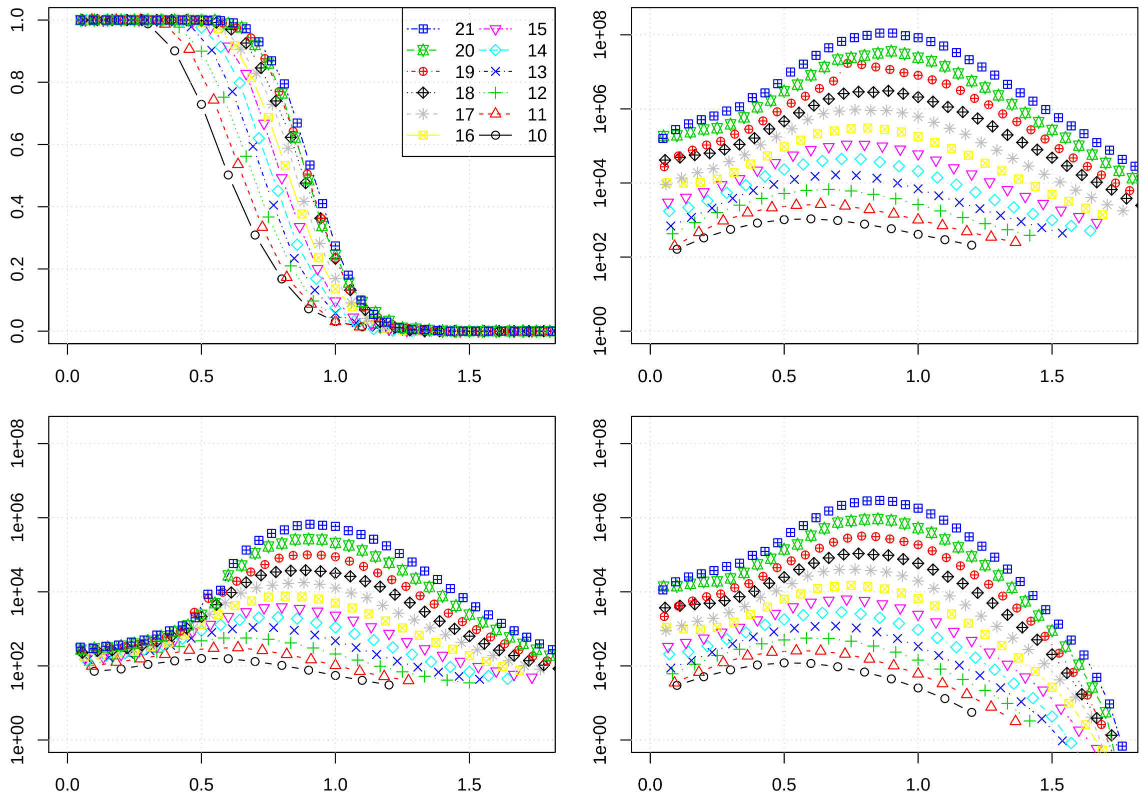


Figure 11: Random instances of the Excluded Diagonals Problem. The x axis represents d/n on all plots, and the colour and plot symbol indicate the value of n. Probability of a solution existing on y axis (upper left), mean nodes searched using special purpose solver (upper right), Lingeling (lower left) and Minion (lower right). The key is the same for all plots and the range of the y axis is the same for all hardness plots.

exclude out of the $4n-2$ possible diagonals (i.e. $2n-1$ each of sum- and difference-diagonals). That is, we construct an instance $M_4 = \langle n, \{\}, \{\}, D^-, D^+\rangle$ subject to $|D^-| + |D^+| = d$. Diagonals are selected one by one, randomly from those remaining, but rejected if excluding the diagonal leaves any row or column with 3 or fewer unblocked squares. For $d > 2n-2$ instances are trivially unsolvable: a solution requires $2n$ allowed diagonals.

Results with our three solver types are shown in Figure 11. We do observe a transition in solvability. It appears to be sharper as n increases. For these sizes the transition in solvability occurs at $d/n \approx 1$, though we cannot say if that trend continues as n increases. All of the solvers show a hardness peak associated with the solvability transition. It is interesting that solving is so hard that we could only experiment up to $n = 21$. The maximum mean time taken for the constraint solver was 262.7s, 78.3s for the SAT solver, but only 4.6s for the special purpose solver. The special purpose solver searches many times more nodes than the more intelligent methods, but it searches so fast that for these instances it is the fastest.

The Excluded Diagonals Problem may be translated to both n-Queens Completion and Blocked n-Queens so can be used to provide hard random instances of both problems. Some interesting questions remain for this generator. We do not know if this generator continues to produce hard instances for larger n, and if so where the phase transition point is. Although the instances can easily be translated to Blocked n-Queens, the definition of κ above is not applicable here because the resulting instances are not drawn from the same random distribution: it would be interesting to find an appropriate definition of κ.

6. Conclusions

We have shown that the n-Queens Completion problem, the Blocked n-Queens problem, and the Excluded Diagonals Problem are NP-Complete and #P-Complete. For anybody who understands the rules of chess, n-Queens Completion may be one of the most natural NP-Complete problems of all: "Given an $n \times n$ chessboard on which some queens are already placed, can you place a queen in every remaining row so that no two queens attack each other?"

We have presented generators for hard random instances of the Blocked n-Queens and Excluded Diagonals problems, shown that this hardness persists across three solvers using very different technologies, and that the hardness is associated with a phase transition in solvability. A natural model for n-Queens Completion does not appear to generate consistently hard instances. We have also shown that care must be taken in generation methods to avoid flaws which can be found in extant benchmarks. Our experiments raise many interesting questions for future research. These include the asymptotic scaling of the transition in Blocked n-Queens and the Excluded Diagonals problems. An interesting open problem is to find a direct generator of hard random instances for n-Queens Completion which works by randomly placing queens on a chessboard without using reductions.

The importance of our work is that the n-Queens problem has been a very widely used, but often criticised, benchmark in Artificial Intelligence. Our work shows that the n-Queens Completion problem or the closely related Blocked n-Queens problem can serve as valid benchmark problems. We implemented three complete methods using different underlying technologies. The best of these in our experiments is to translate into SAT using the commander encoding and solve with a modern SAT solver, but it would be interesting to compare this with more of the very many techniques that have been applied to the n-Queens problem. Solvers for n-Queens typically require minimal adaptation to work on the variants in this paper. By providing random instance generators we hope to encourage researchers to test their n-Queens algorithms on variants that can better discriminate between good and bad solvers than the original problem.

Acknowledgments

We thank Jeff Erickson for raising the question of the complexity of (a slight variant of) n-Queens Completion (Erickson, 2014) and Stanislav Živný for bringing his post to our attention by a Facebook post on 1 January 2015. We thank Torsten Schaub, Martin Gebser, Bilal Hussain, and András Salamon. We would like to thank the EPSRC for partially

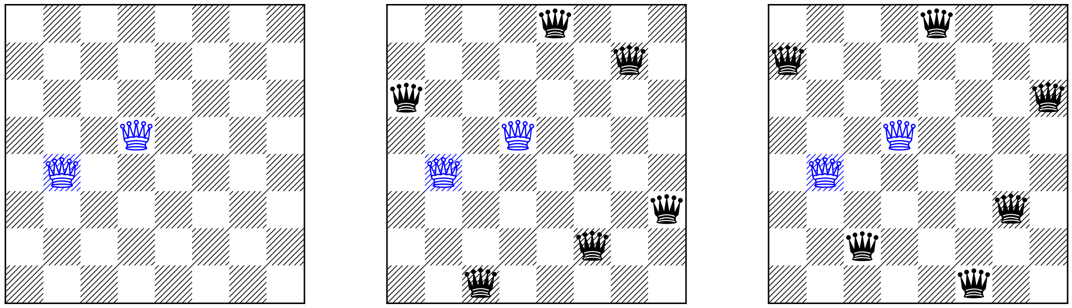


Figure 12: The 8-Queens Completion instance from Figure 1 and its two possible solutions.

funding this work through grants EP/K015745/1, EP/M003728/1, and EP/P015638/1. In addition, Dr Jefferson is funded by a Royal Society University Research Fellowship. We thank the JAIR reviewers and associate editor for their diligent work which led to significant improvements in the paper.

References

Achlioptas, D. (2009). Random satisfiability. In Biere et al. (Biere, Heule, van Maaren, & Walsh, 2009), pp. 245–270.

Achlioptas, D., Molloy, M. S. O., Kirousis, L. M., Stamatiou, Y. C., Kranakis, E., & Krizanc, D. (2001). Random constraint satisfaction: A more accurate picture. *Constraints*, *6*(4), 329–344.

Altarelli, F., Monasson, R., Semerjian, G., & Zamponi, F. (2009). Connections to statistical physics. In Biere et al. (Biere et al., 2009), pp. 569–611.

Bell, J., & Stevens, B. (2009). A survey of known results and research areas for n-queens. *Discrete Mathematics*, *309*(1), 1 – 31.

Bernhardsson, B. (1991). Explicit solutions to the n-queens problem for all n. *SIGART Bulletin*, *2*(2), 7.

Bezzel, M. (1848). Schachfreund. *Berliner Schachzeitung*, *3*, 363. Cited in (Campbell, 1977).

Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.). (2009). *Handbook of Satisfiability*, Vol. 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Bitner, J. R., & Reingold, E. M. (1975). Backtrack programming techniques. *Communications of the ACM*, *18*(11), 651–656.

Cadoli, M., & Schaerf, M. (2006). Partial solutions with unique completion. In Stock, O., & Schaerf, M. (Eds.), *Reasoning, Action and Interaction in AI Theories and Systems, Essays Dedicated to Luigia Carlucci Aiello*, Vol. 4155 of *Lecture Notes in Computer Science*, pp. 101–115. Springer.

Campbell, P. J. (1977). Gauss and the eight queens problem: A study in miniature of the propagation of historical error. *Historia Mathematica*, *4*(4), 397 – 404.

Chaiken, S., Hanusa, C., & Zaslavsky, T. (2015). A q-queens problem. ii. the square board. *Journal of Algebraic Combinatorics*, *41*(3), 619–642.

Cheeseman, P., Kanefsky, B., & Taylor, W. M. (1991). Where the Really Hard Problems Are. In *Proc. 12th International Joint Conference on Artificial Intelligence (IJCAI 91)*, pp. 331–337.

Cloitre, B. (2002). Strong conjecture. In Sloane, N. J. A. (Ed.), *The On-Line Encyclopedia of Integer Sequences, Sequence A000170*.

Colbourn, C. J. (1984). The complexity of completing partial latin squares. *Discrete Applied Mathematics*, *8*, 25–30.

Crawford, J. M., Ginsberg, M. L., Luks, E. M., & Roy, A. (1996). Symmetry-breaking predicates for search problems. In Aiello, L. C., Doyle, J., & Shapiro, S. C. (Eds.), *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96), Cambridge, Massachusetts, USA, November 5-8, 1996.*, pp. 148–159. Morgan Kaufmann.

Creignou, N., & Hermann, M. (1996). Complexity of generalized satisfiability counting problems. *Information and Computation*, *125*(1), 1 – 12.

Demaine, E. D., & Hearn, R. A. (2009). Playing games with algorithms: algorithmic combinatorial game theory. In *Games of No Chance 3*, pp. 3–56. Cambridge University Press.

Erickson, J. (2014). Complexity of n-queens-completion?. Theoretical Computer Science Stack Exchange. URL:http://cstheory.stackexchange.com/q/28002 (version: 2014-12-29).

Fraenkel, A. S., & Lichtenstein, D. (1981). Computing a perfect strategy for $n \times n$ chess requires time exponential in n. *Journal of Combinatorial Theory, Series A*, *31*(2), 199 – 214.

Gebser, M., Liu, L., Namasivayam, G., Neumann, A., Schaub, T., & Truszczyński, M. (2007). The first answer set programming system competition. In *Proceedings of the International Conference on Logic Programming and Nonmonotonic Reasoning*, pp. 3–17.

Gent, I., Jefferson, C., & Miguel, I. (2006). Minion: A fast, scalable, constraint solver. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006)*, pp. 98–102.

Gent, I. P., Jefferson, C., Kelsey, T., Lynce, I., Miguel, I., Nightingale, P., Smith, B. M., & Tarim, A. (2007). Search in the patience game 'Black Hole'. *AI Communications*, *20*(3), 211–226.

Gent, I. P., MacIntyre, E., Prosser, P., & Walsh, T. (1995). Scaling effects in the CSP phase transition. In *Proc. First International Conference on the Principles and Practice of Constraint Programming (CP 95)*, pp. 70–87.

Gent, I. P., MacIntyre, E., Prosser, P., & Walsh, T. (1996). The constrainedness of search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and*

Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, August 4-8, 1996, Volume 1, pp. 246–252.

Gent, I. P., Nightingale, P., Rowley, A., & Stergiou, K. (2008). Solving quantified constraint satisfaction problems. *Artificial Intelligence*, *172*(6-7), 738–771.

Gent, I. P., & Walsh, T. (1994). Easy problems are sometimes hard. *Artificial Intelligence*, *70*, 335–345.

Gent, I. P., & Walsh, T. (1999). Beyond NP: the QSAT phase transition. In Hendler, J., & Subramanian, D. (Eds.), *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence, July 18-22, 1999, Orlando, Florida, USA.*, pp. 648–653. AAAI Press / The MIT Press.

Golomb, S. W., & Baumert, L. D. (1965). Backtrack programming. *Journal of the ACM*, *12*(4), 516–524.

Gomes, C. P., & Selman, B. (1997). Problem structure in the presence of perturbations. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI 97)*, pp. 221–226.

Gu, J. (1991). On a general framework for large-scale constraint-based optimization. *SIGART Bulletin*, *2*(2), 8.

Guala, L., Leucci, S., & Natale, E. (2014). Bejeweled, Candy Crush and other match-three games are (NP-) hard. In *2014 IEEE Conference on Computational Intelligence and Games*.

Helmert, M. (2003). Complexity results for standard benchmark domains in planning. *Artificial Intelligence*, *143*(2), 219–262.

Hogg, T., Huberman, B. A., & Williams, C. P. (1996). Phase transitions and the search problem. *Artificial Intelligence*, *81*(1), 1 – 15.

Hogg, T., & Williams, C. P. (1994). The hardest constraint problems: a double phase transition. *Artificial Intelligence*, *69*, 359–377.

Hsiang, J., Hsu, D., & Shieh, Y.-P. (2004). On the hardness of counting problems of complete mappings. *Discrete Mathematics*, *277*(1–3), 87 – 100.

Järvisalo, M., Heule, M. J., & Biere, A. (2012). Inprocessing rules. In *Automated Reasoning, 6th International Joint Conference, IJCAR 2012*, pp. 355–370. Springer.

Johnson, W. L. (1991). Letter from the editor. *SIGART Bulletin*, *2*(2), 1.

Kaye, R. (2000). Minesweeper is NP-complete. *The Mathematical Intelligencer*, *22*(2), 9–15.

Kendall, G., Parkes, A. J., & Spoerer, K. (2008). A survey of NP-complete puzzles. *ICGA Journal*, *31*(1), 13–34.

Klieber, W., & Kwon, G. (2007). Efficient CNF encoding for selecting 1 from n objects. In *Proc. International Workshop on Constraints in Formal Verification.*

Longpré, L., & McKenzie, P. (2009). The complexity of solitaire. *Theoretical Computer Science*, *410*(50), 5252–5260.

Mackworth, A. K., & Freuder, E. C. (1985). The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, *25*(1), 65–74.

Mandziuk, J. (1995). Solving the n-queens problem with a binary Hopfield-type network. synchronous and asynchronous model. *Biological Cybernetics*, *72*(5), 439–446.

Martin, B. (2007). On the complexity of a derivative chess problem. *CoRR*, *arXiv:cs/0701049 [cs.CC]*.

Martinjak, I., & Golub, M. (2007). Comparison of heuristic algorithms for the n-queen problem. In *2007 29th International Conference on Information Technology Interfaces*, pp. 759–764.

Minton, S., Johnston, M. D., Philips, A. B., & Laird, P. (1992). Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, *58*(1-3), 161–205.

Mitchell, D. G., Selman, B., & Levesque, H. J. (1992). Hard and easy distributions of SAT problems. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92)*, pp. 459–465.

Nakaguchi, T., Kenya, J., & Tanaka, M. (1999). Hysteresis neural networks for n-queens problems. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, *82*(9), 1851–1859.

Namasivayam, G., & Truszczynski, M. (2009). Blocked n-queens. `https://dtai.cs.kuleuven.be/events/ASP-competition/Benchmarks/BlockedQueens.shtml`.

Nauck, F. (1850). Schach: Eine in das Gebiet der Mathematik fallende Aufgabe von Herrn Dr. Nauck in Schleusingen. *Illustrirte Zeitung*, *14*(361), 352. Cited in (Campbell, 1977).

Nightingale, P., Akgün, O., Gent, I. P., Jefferson, C., Miguel, I., & Spracklen, P. (2017). Automatically improving constraint models in Savile Row. *Artificial Intelligence*, *251*, 35–61.

Papadimitriou, C. H. (1994). *Computational complexity*. Addison-Wesley.

Porschen, S., Schmidt, T., Speckenmeyer, E., & Wotzlaw, A. (2014). XSAT and NAE-SAT of linear CNF classes. *Discrete Applied Mathematics*, *167*, 1 – 14.

Preußer, T. B. (2016). 27-queens puzzle: Massively parellel enumeration and solution counting. `https://github.com/preusser/q27`.

Richards, M. (1997). Backtracking algorithms in MCPL using bit patterns and recursion. Tech. rep., Computer Laboratory, University of Cambridge.

Rivin, I., & Zabih, R. (1992). A dynamic programming solution to the n-queens problem. *Information Processing Letters*, *41*(5), 253–256.

Selman, B., Levesque, H. J., & Mitchell, D. G. (1992). A new method for solving hard satisfiability problems. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92)*, pp. 440–446.

Shah-Hosseini, H. (2009). The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. *International Journal of Bio-Inspired Computation*, *1*(1-2), 71–79.

Sloane, N. J. A. (2016). The on-line encyclopedia of integer sequences. `https://oeis.org`.

Smet, G. D. (2014). Cheating on the n queens benchmark. `http://www.optaplanner.org/blog/2014/05/12/CheatingOnTheNQueensBenchmark.html`.

Sosic, R., & Gu, J. (1990). A polynomial time algorithm for the n-queens problem. *SIGART Bulletin*, *1*(3), 7–11.

Takayuki, Y., & Takahiro, S. (2003). Complexity and completeness of finding another solution and its application to puzzles. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, *86*(5), 1052–1060.

Valtorta, M. (1991). Response to "explicit solutions to the n-queens problem for all n". *SIGART Bulletin*, *2*(4), 7–5.

Viglietta, G. (2014). Gaming is a hard job, but someone has to do it!. *Theory of Computing Systems*, *54*(4), 595–621.

Walker, R. J. (1960). An enumerative technique for a class of combinatorial problems. In Bellmann, R., & Jr, M. H. (Eds.), *Combinatorial Analysis: Proceedings Of The Tenth Symposium In Applied Mathematics Of The American Mathematical Society*, pp. 91–94. American Mathematical Society.

Walsh, T. (2014). Candy Crush is NP-hard. *CoRR*, *arXiv:1403.1911 [cs.CC]*.